

DTIC FILE COPY

2

GL-TR-89-0307

POLAR User's Manual

**John R. Lilley, Jr.
David L. Cooke
Gary A. Jongeward
Ira Katz**

**Maxwell Laboratories, Inc.
S-CUBED Division
P. O. Box 1620
La Jolla, CA 92038-1620**

October 1989

Scientific Report No. 9

Approved for public release; distribution unlimited

**Geophysics Laboratory
Air Force Systems Command
United States Air Force
Hanscom Air Force base, Massachusetts 01731-5000**

**DTIC
ELECTE
FEB 12 1991
S B D**

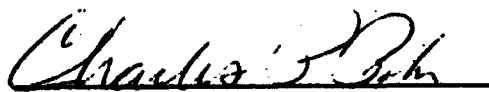
91 2 11 174

AD-A232 103

" This technical report has been reviewed and is approved for publication "



DAVID L. COOKE
Contract Manager



CHARLES P. PIKE
Branch Chief

FOR THE COMMANDER



RITA C. SAGALYN
Division Director

This report has been reviewed by the ESD Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS).

Qualified requestors may obtain additional copies from the Defense Technical Information Center. All others should apply to the National Technical Information Services.

If your address has changed, or if you wish to be removed from the mailing list, or if the addressee is no longer employed by your organization, please notify AFGL/DAA, Hanscom AFB, MA 01731. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
1a. REPORT SECURITY CLASSIFICATION Unclassified			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) SSS-R-86-7563/R2			5. MONITORING ORGANIZATION REPORT NUMBER(S) GL-TR-89-0307		
6a. NAME OF PERFORMING ORGANIZATION S-CUBED Division Maxwell Laboratories, Inc.		6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Geophysics Laboratory		
6c. ADDRESS (City, State, and ZIP Code) P. O. Box 1620 La Jolla, CA 92038-1620			7b. ADDRESS (City, State, and ZIP Code) Hanscom AFB MA 01731-5000		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION		8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F19628-86-C-0056		
8c. ADDRESS (City, State, and ZIP Code)			10. SOURCE OF FUNDING NUMBERS		
			PROGRAM ELEMENT NO. 62101F	PROJECT NO. 7601	TASK NO. 30
					WORK UNIT ACCESSION NO. AA
11. TITLE (Include Security Classification) POLAR USER'S MANUAL					
12. PERSONAL AUTHOR(S) John R. Lilley, Jr.; David L. Cooke; Gary A. Jongeward; Ira Katz					
13a. TYPE OF REPORT Scientific #9		13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Year, Month, Day) 1989 October		15. PAGE COUNT 474
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	Spacecraft charging		
			Finite elements		
			Auroral ionosphere		
			3-D		
			Poisson solution		
			POLAR computer code		
19. ABSTRACT (Continue on reverse if necessary and identify by block number) This report documents the physical principles and computational algorithms of the POLAR code. POLAR models in three dimensions the interactions of large spacecraft with the plasma environment in low polar orbit. It includes models of space charge limited particle collection, satellite wakes, the polar auroral environment, magnetic field effects, spacecraft surface charging, particle beam effects, and sheath ionization.					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION Unclassified		
22a. NAME OF RESPONSIBLE INDIVIDUAL David L. Cooke			22b. TELEPHONE (Include Area Code) (617) 377-2931		22c. OFFICE SYMBOL PHK

TABLE OF CONTENTS

<u>Chapter</u>		<u>Page</u>
	LIST OF ILLUSTRATIONS.....	iii
	LIST OF TABLES	xii
1.	INTRODUCTION	1.1-1
	1.10 CODE STRUCTURE.....	1.1-1
	1.20 DOCUMENTATION.....	1.2-1
2.	THE PHYSICS OF LARGE STRUCTURES IN THE POLAR IONOSPHERE	2.0-1
3.	PHYSICAL MODELS EMPLOYED IN THE POLAR CODE.....	3.1-1
	3.10 THE POLAR PLASMA ENVIRONMENT.....	3.1-1
	3.20 PLASMA POTENTIALS.....	3.1-2
	3.30 PARTICLE DENSITIES.....	3.3-1
	3.31 STRUCTURE OF THE PLASMA WAKE.....	3.3-4
	3.32 SHEATH DENSITIES.....	3.3-8
	3.40 SURFACE CURRENTS.....	3.4-1
	3.41 ANALYTICAL ELECTRON SURFACE CURRENTS..	3.4-2
	3.42 ATTRACTED PARTICLE SURFACE CURRENTS...	3.4-4
	3.43 INITIAL SHEATH PARTICLE VELOCITY DISTRIBUTIONS.....	3.4-6
	3.50 ELECTRICAL CHARGING.....	3.5-1
	3.60 THE POLAR SHEATH MODEL.....	3.6-1
	REFERENCES, CHAPTER 3.....	3.6-3
4.	COMPUTATIONAL TECHNIQUES.....	4.1-1
	4.10 GRIDS - DISCRETIZATION OF SPACE.....	4.1-1
	4.11 STAGGERED MESH.....	4.1-1
	4.12 OBJECT GRID.....	4.1-2
	4.20 POTENTIAL CALCULATIONS.....	4.2-1
	4.21 FINITE ELEMENTS.....	4.2-1



Codes
/or

TABLE OF CONTENTS (CONTINUED)

<u>Chapter</u>		<u>Page</u>
4.21.1	GENERAL.....	4.2-1
4.21.2	SPECIAL CELLS.....	4.2-5
4.21.21	INTERPOLATION FUNCTIONS FOR FACE-CENTERED SURFACE NODES (FCSN'S).....	4.2-6
4.21.22	THE EMPTY CUBE (TYPE 0) ELEMENT WITH NO FCSN'S.....	4.2-7
4.21.23	THE EMPTY CUBE (TYPE 0) ELEMENT WITH SIX FCSN'S.....	4.2-8
4.21.24	THE WEDGE ELEMENT (TYPE 1)...	4.2-13
4.21.25	THE TYPE 2 ELEMENT.....	4.2-17
4.21.26	THE TETRAHEDRON ELEMENT.....	4.2-21
4.21.27	THE TRUNCATED CUBE ELEMENT (TYPE 4).....	4.2-24
4.21.28	THE SLANTED THIN PLATE ELEMENT (TYPE 5).....	4.2-27
4.22	BOUNDARY CONDITIONS.....	4.2-27
4.30	MATRIX SOLVERS.....	4.3-1
4.31	CONJUGATE GRADIENT METHOD.....	4.3-1
4.32	ICCG, THE INCOMPLETE CHOLSKY CONJUGATE GRADIENT METHOD.....	4.3-3
4.40	SPACE CHARGE AND CURRENT COMPUTATION.....	4.4-1
4.41	WEAK FIELD IONS, PRESHEATH AND WAKE...	4.4-1
4.42	THE POLAR SHEATH MODEL (TECHNICAL).....	4.4-2
4.42.1	SHEATH EDGE ALGORITHM (SHEATH).....	4.4-2
4.42.2	CURRENTS TO THE SHEATH SURFACE.....	4.4-2
4.42.3	SHEATH PARTICLE ASSIGNMENT...	4.4-8
4.42.4	TRAJECTORY TRACKING.....	4.4-9

TABLE OF CONTENTS (CONTINUED)

<u>Chapter</u>		<u>Page</u>
	4.42.5 SHEATH ION DENSITIES.....	4.4-11
4.43	CHARGE DENSITY.....	4.4-12
	4.43.1 ELECTRONS.....	4.4-13
	4.43.2 ION CHARGE DENSITY.....	4.4-13
4.44	THE CHARGE STABILIZED POISSON ITERATION.....	4.4-15
	4.44.1 SHEATH IONIZATION EFFECTS ON SPACE CHARGE.....	4.4-18
	4.44.2 ANALYSIS OF THE CHARGE STABILIZED POISSON METHOD . .	4.4-22
	4.44.3 PARTICLE BEAM SPACE CHARGE EFFECTS.....	4.4-27
	4.44.4 ANALYTIC FORMULATION FOR SHEATH CONVERGENCE.....	4.4-27
4.50	CHARGING MODEL.....	4.5-1
4.51	CONDUCTOR CURRENTS AND CURRENT DERIVATIVES.....	4.5-3
4.52	ELECTRON CURRENTS, PRIMARY, SECONDARY.....	4.5-6
	4.52.1 SECONDARY ELECTRONS.....	4.5-6
	4.52.2 BACKSCATTER ELECTRONS.....	4.5-12
	4.52.3 INTEGRAL OF THE MAXWELLIAN DISTRIBUTION.....	4.5-13
	4.52.4 INTEGRAL OF THE POWER LAW DISTRIBUTION.....	4.5-15
	4.52.5 INTEGRAL OF THE GAUSSIAN DISTRIBUTION ELECTRONS.....	4.5-17
	4.52.6 PHOTOEMISSION.....	4.5-19
	4.52.7 SHEATH TO OBJECT ELECTRON CURRENTS.....	4.5-20
4.53	ION SURFACE CURRENTS.....	4.5-22

TABLE OF CONTENTS (CONTINUED)

<u>Chapter</u>		<u>Page</u>
	4.53.10 THERMAL ION SURFACE CURRENTS.....	4.5-23
	4.53.20 SHEATH TO OBJECT ION CURRENTS.....	4.5-25
	4.53.30 ION CURRENT DERIVATIVE.....	4.5-30
	4.54 SURFACE INTERACTIONS.....	4.5-32
	4.54.10 SURFACE CONDUCTIVITY.....	4.5-32
	4.54.20 PHOTOCONDUCTION/HOPPING SECONDARIES.....	4.5-33
	4.54.30 SURFACE TO PLASMA CAPACITANCE.....	4.5-34
	4.54.40 SURFACE TO CONDUCTOR CAPACITANCE.....	4.5-34
	4.55 CIRCUIT MODEL.....	4.5-36
	4.56 CHARGING ALGORITHM.....	4.5-39
	4.56.10 CHARGING NOTATION.....	4.5-39
	4.56.20 DETAILS OF THE CHARGING ALGORITHM.....	4.5-42
	4.56.30 PARTICLE BEAM CHARGING EFFECTS.....	4.5-46
	4.57 CHARGING MATRIX FORMULATIONS.....	4.5-47
	4.57.10 AN EXAMPLE OF MATRIX FORMULATION.....	4.5-50
	REFERENCES, CHAPTER 4.....	4.5-57
5.	POLAR CODE STRUCTURE	5.1-1
	5.10 TOP DOWN VIEW OF THE POLAR PACKAGE...	5.1-1
	5.11 VEHICL	5.1-3
	5.12 ORIENT	5.1-5
	5.13 NTERAK	5.1-6
	5.14 SHONTL	5.1-8

TABLE OF CONTENTS (CONTINUED)

<u>Chapter</u>	<u>Page</u>
5.15 UTILITIES.....	5.1-8
5.20 SLICE GRID SYSTEM.....	5.2-1
5.21 SLICE MACHINERY.....	5.2-4
5.22 VOLUME ELEMENT MACHINERY.....	5.2-7
5.23 ELEMENT TABLE, LTBL.....	5.2-9
5.24 SURFACE CELLS.....	5.2-11
5.24.1 SURFACE CELL LIST, KSURF....	5.2-11
5.25 LCEL, CONNECTIVITY.....	5.2-14
5.30 FILE SYSTEM.....	5.3-1
5.31 MASS STORAGE FILE MANAGEMENT.....	5.3-2
5.32 MRBUF PLUS BUFSET PLUS FRIENDS.....	5.3-4
5.33 MRBUF VARIABLE LIST.....	5.3-7
5.40 OBJECT DEFINITION.....	5.4-1
5.50 POTENTIALS.....	5.5-1
5.60 PARTICLE DENSITIES.....	5.6-1
5.61 PRESHEATH SPACE CHARGE DENSITIES	5.6-1
5.61.10 NEUTRAL ION APPROXIMATION (NEUDEN).....	5.6-1
5.61.15 SHADO APPROACH.....	5.6-6
5.61.16 SHADO STRUCTURE.....	5.6-10
5.61.20 ELECTRIC FIELD CORRECTION FOR NEUTRAL IONS.....	5.6-17
5.62 SHEATH PARTICLES.....	5.6-19
5.62.10 SHEATH EDGE.....	5.6-20
5.62.11 THE PARTICLE LIST STRUCTURE.....	5.6-22
5.62.12 PARTICLE PUSHING UNITS.....	5.6-25
5.62.20 SHEATH CURRENT.....	5.6-26

TABLE OF CONTENTS (CONTINUED)

<u>Chapter</u>		<u>Page</u>
	5.62.21 CURPEP (CURRENT PREPARER)...	5.6-27
	5.62.22 PUSHER (PARTICLE PUSHING)...	5.6-28
	5.62.23 SHEATH PARTICLE DENSITY.....	5.6-32
	5.62.24 CUEXIT (CURRENT EXIT ROUTINE).....	5.6-32
5.70	SURFACE CHARGING.....	5.7-1
	5.71 PUSHED PARTICLE SURFACE CURRENTS.....	5.7-1
	5.72 CHARGE MODES.....	5.7-3
	5.73 CHARGE (SURFACE CHARGING CONTROL)....	5.7-3
	5.73.1 SURCHG (SURFACE CHARGER)....	5.7-5
	5.73.2 CHARGING MATRIX AND VECTOR FORMULATION.....	5.7-11
5.80	OUTPUT	5.8-1
	5.81 GENERAL	5.8-1
	5.82 GRAPHICAL CODE STRUCTURE.....	5.8-1
	5.82.10 AN OVERVIEW OF SHONTL.....	5.8-1
	5.82.11 SHONTL.....	5.8-3
	5.82.12 SHODEF (PLOT INITIALIZATION).....	5.8-3
	5.82.13 SHOINP (SHONTL INPUT).....	5.8-4
	5.82.14 GENPLT (GENERATE PLOTS).....	5.8-4
	5.82.15 SHOXIT (SHONTL EXIT).....	5.8-4
	REFERENCES, CHAPTER 5.....	5.8-5
6.	OPERATING INSTRUCTIONS.....	6.1-0
	6.10 OBJECTS	6.1-1
	6.10.10 BUILDING BLOCKS.....	6.1-4
	6.10.11 COMMANDS (OR HOW DO I ACTUALLY DEFINE AN OBJECT).....	6.1-4

TABLE OF CONTENTS (CONTINUED)

<u>Chapter</u>		<u>Page</u>
	6.10.12 PLATES AND PATCHES.....	6.1-7
	6.10.13 SPECIAL SHAPES.....	6.1-9
	6.10.14 BUILDING BLOCK PARAMETERS (OR WHO'S ON NEXT).....	6.1-9
	6.10.15 RECTAN.....	6.1-11
	6.10.16 PATCHR.....	6.1-13
	6.10.17 WEDGE.....	6.1-13
	6.10.18 PATCHW.....	6.1-17
	6.10.19 TETRAH.....	6.1-17
	6.10.20 OCTAGON.....	6.1-19
	6.10.21 QSPHERE.....	6.10-24
	6.10.22 FIL111.....	6.1-26
	6.10.23 PLATE	6.1-28
	6.10.24 SLANT	6.1-30
	6.10.25 MORE OBJECT DEFINITION KEYWORDS.....	6.1-30
6.11	DEFINING AN OBJECT: AN EXAMPLE.....	6.1-34
	6.11.10 LIMITATIONS IN OBJECT DEFINITION.....	6.1-37
	6.11.11 DOUBLE POINTS.....	6.1-37
	6.11.12 TRIPLE POINTS.....	6.1-39
6.12	SURFACE MATERIALS.....	6.1-41
	6.12.10 MATERIAL PROPERTIES.....	6.1-41
	6.12.11 DEFINING MATERIALS.....	6.1-46
	6.12.12 DEFAULT MATERIALS.....	6.1-49
6.13	THE OBJECT DEFINITION FILE - ANOTHER EXAMPLE.....	6.1-57
6.14	OBJECTS WITHIN OBJECTS: VARIEGATED SURFACES.....	6.1-60

TABLE OF CONTENTS (CONTINUED)

<u>Chapter</u>		<u>Page</u>
6.20	VEHICL	6.2-1
6.21	VEHICL KEYWORDS.....	6.2-1
6.22	VEHICL DIAGNOSTIC KEYWORDS.....	6.2-10
6.23	AN EXAMPLE OF A VEHICL RUN.....	6.2-12
6.24	TROUBLESHOOTING VEHICL.....	6.2-13
6.30	ORIENT	6.3-1
6.31	ORIENT KEYWORDS.....	6.3-1
6.32	RUNNING ORIENT.....	6.3-4
6.40	NTERAK	6.4-1
6.41	NTERAK CONTROL KEYWORDS.....	6.4-2
6.42	KEYWORDS TO SET UP NTERAK.....	6.4-10
6.42.10	PLASMA ENVIRONMENT.....	6.4-10
6.42.20	MAGNETIC FIELDS.....	6.4-18
6.42.30	NEUTRAL ION DENSITY.....	6.4-19
6.42.40	INITIAL VOLTAGES AND ELECTRIC MODEL.....	6.4-26
6.42.50	GRID SIZE CONTROL.....	6.4-33
6.42.60	PARTICLE BEAMS.....	6.4-35
6.42.70	SHEATH IONIZATION.....	6.4-38
6.43	NTERAK SUBSECTION CONTROL.....	6.4-39
6.43.10	PWASON (POISSON POTENTIAL SOLVER).....	6.4-40
6.43.20	CURREN (ION CURRENT CAL- CULATION).....	6.4-46
6.43.30	CHARGE (SURFACE CHARGER)....	6.4-50
6.44	NTERAK OUTPUT CONTROL.....	6.4-59
6.44.10	CALCULATION MONITORING.....	6.4-60
6.44.20	DIAGNOSTIC OUTPUT.....	6.4-65

TABLE OF CONTENTS (CONCLUDED)

<u>Chapter</u>		<u>Page</u>
6.45	SUMMARY OF NTERAK KEYWORDS.....	6.4-75
6.45.10	NTERAK DIAGNOSTICS AND OUTPUT CONTROL.....	6.4-86
6.50	OPERATING SHONTL.....	6.5-1
6.51	STEP BY STEP INSTRUCTIONS FOR SHONTL	6.5-3
6.52	SHONTL KEYWORDS.....	6.5-4
6.53	SPECTRUM KEYWORDS AND OPERATING INSTRUCTIONS.....	6.5-12
6.54	SHONTL DEFAULTS.....	6.5-13
6.55	SPECIAL SHONTL OPTIONS.....	6.5-13
6.60	PLOTTING UTILITIES.....	6.6.1
6.70	SURFACE CHANGING UTILITY.....	6.7-1
6.80	TRMTLK-NTERAK RUN ANALYSIS TOOL.....	6.8-1
6.81	PROGRAM STRUCTURE.....	6.8-2
6.82	CELL SPECIFICATIONS.....	6.8-2
6.83	CHARGING HISTORY.....	6.8-2
6.84	INSTRUCTIONS FOR USE.....	6.8-3
6.85	SAMPLE RUN.....	6.8-3
6.86	INTERNAL DOCUMENTATION.....	6.8-8
6.90	SOURCE CODE MAINTENANCE.....	6.9-1
6.91	INSTALLING THE SOURCE CODE.....	6.9-2
6.92	GENERAL INFORMATION FOR THE MAIN SOURCE CODE VERSION.....	6.9-5
6.93	INFORMATION FOR MAKING LOCAL OR USER MODIFIED VERSIONS OF POLAR.....	6.9-10
6.94	A QUICK SUMMARY OF MAKEFILE SYSTEM...	6.9-15

LIST OF ILLUSTRATIONS

<u>Figure No.</u>		<u>Page</u>
3.30/1	The central dimensionless potential ($\Phi = eV/kT$) of a cylindrical ion void of radius R.....	3.3-2
3.31/1	Different wake code interest regions.....	3.3-4
3.41/1	Pitch angle and the spherical polar angles.....	3.4-3
4.11/1	(A) Cube moving to left. (B) Rising quasi-sphere. (C) Falling quasisphere.....	4.1-3
4.21.23/1	Cubical finite elements with six face-centered surface nodes (FCSNs).....	4.2-8
4.21.23/2	Interpolation functions for the cubical element of Figure 4.21.23/1.....	4.2-9
4.42/1	Coordinate systems.....	4.4-3
4.44/1	Plots of space charge.....	4.4-25
4.44/2	Plot of space charge cutoff potential.....	4.4-26
4.52/1a	Energy deposition profiles.....	4.5-8
4.52/1b	Generalized yield curve.....	4.5-8
4.52/2	Secondary emission be aluminum.....	4.5-11
4.53/1	Current sharing on a quasisphere.....	4.5-26
4.53/2	Bilinear weighting of triangular surfaces.....	4.5-26
4.53/3	Bilinear weight ($w_{b_{sa}}$) of a rectangular surface.....	4.5-27
4.55/1	Legend of circuit elements.....	4.5-37
4.55/2	Circuit representation of a single insulating surface.....	4.5-38
4.55/3	Circuit representation of a single exposed conductor surface.....	4.5-38
4.55/4	Circuit representation of two insulating surfaces with a common underlying conductor.....	4.5-38
4.56/1	A multiple crossover I-V curve.....	4.5-44
4.57/1	General circuit model used to study matrix construction.....	4.5-50

LIST OF ILLUSTRATIONS (CONTINUED)

<u>Figure No.</u>		<u>Page</u>
5.11/1	VEHICL structure.....	5.1-3
5.12/1	ORIENT structure.....	5.1-5
5.13/1	Structure of NTERAK module.....	5.1-6
5.20/1	An x-z cut of a typical NTERAK computational mesh.....	5.2-2
5.24/1	KSURF surface cell list bit code.....	5.2-13
5.50/1	PWASON structure.....	5.5-1
5.50/2	CONGRD structure.....	5.5-2
5.61/1	Neutral approximation phase space map of a flying brick blocking ram direction.....	5.6-3
5.61/2	Neutral approximation phase space map of a flying brick at right angles to ram direction....	5.6-4
5.61/3	Flying brick in anti-ram direction.....	5.6-5
5.61/4	SHADO structure program.....	5.6-11
5.61/5	SHDSET structure diagram.....	5.6-12
5.61/6	Structure diagram for the SHADO subroutine.....	5.6-16
5.62.20/1	Structure diagram of subroutine CURREN.....	5.6-26
5.71.1	Structure diagram of IONCUR segment.....	5.7-1
5.73/1	Structure diagram of CHARGE module.....	5.7-4
5.73/2	Structure diagram of routine SURCHG.....	5.7-5
5.82/1	A general structure diagram of SHONTL.....	5.8-3
6/1	Cuboid made by filling in twenty-four volume elements.....	6.1-2
6/2	Four shapes of volume cells considered by the POLAR code.....	6.1-3
6/3	Eight building block types.....	6.1-5
6/4a	A FIL111 building block.....	6.1-10
6/4b	"Steps" along grid lines.....	6.1-10
6/4c	"Steps" along 456/4c.....	6.1-10

LIST OF ILLUSTRATIONS (CONCLUDED)

<u>Figure No.</u>		<u>Page</u>
6/5	RECTAN.....	6.1-12
6/6	Wedge defined with surface normal 110 and corner 0,0,0.....	6.1-16
6/7	Tetrahedron defined with its "corner" at 000 and a surface normal 111.....	6.1-20
6/8	Top of an OCTAGON.....	6.1-22
6/9	OCTAGON.....	6.1-23
6/10	QSPHERE.....	6.1-25
6/11	FIL111.....	6.1-27
6/12	PLATE.....	6.1-29
6/13	Object definition example.....	6.1-35
6/14	Three views of object defined by input of Figure 6/13.....	6.1-36
6/15	Profile of two cuboids sharing a common edge and resultant double points.....	6.1-38
6/16	Examples of plates intersecting objects.....	6.1-40
6/17	General form of the object definition file.....	6.1-48
6/18	Object definition file.....	6.1-58
6/19	3-D view of object produced by HIDCEL (hidden lines).....	6.1-59
6/20	A variegated surface definition.....	6.1-61
6.2/1	VEHICL runstream.....	6.2-12
6.3/1	Sample ORIENT runstream.....	6.3-4
6.4/1	Comparison of POLAR spectrum and DMSP data.....	6.4-15

LIST OF TABLES

<u>Table No.</u>		<u>Page</u>
5.62.11/1	Example of Particle List Data Structure.....	5.6-23
6/1	POLAR Building Blocks and Their Keywords.....	6.1-6
6/2	Object Definition - File 20.....	6.1-8
6/3	Directions of Surface Normals Associated with Allowed Wedge Orientation.....	6.1-15
6/4	Directions of Surface Normals Associated with Allowed Tetrahedron Orientations.....	6.1-18
6/5	Material Properties.....	6.1-42
6/6	Material Properties.....	6.1-50
6.21/1	Summary of VEHICL keywords.....	6.2-9
6.31/1	Summary of ORIENT keywords.....	6.3-2
6.42/1a	Beam Characteristics - Keyword Definition.....	6.4-36
6.42/1b	Beam Characteristics - Input Syntax.....	6.4-36
6.44/1	Flag Settings for Subsections by Quantity Level.....	6.4-66
6.70/1	Orbit limited collection.....	6.7-1
6.70/2	Space charge limited collection.....	6.7-2

1. INTRODUCTION

POLAR is a set of computer programs designed to predict the electrical interactions between the natural environment and a large spacecraft in polar earth orbit. POLAR consists of many complex physical models which have been converted to algorithms and connected by an executive structure. Since there are a wide variety of spacecraft and environments, POLAR has been written with maximum flexibility and applicability in mind. However, to allow for when a model may prove inadequate, POLAR has been designed with a high degree of modularity to enable changes in the physical models and algorithms to be made quickly and reliably. The documentation is also designed modularly so that code modifications can be documented immediately. Thus this manual is intended to be a living document, accurately reflecting the most current status of the POLAR code. This modularity does make for difficult reading, but we feel that the total information content is enhanced and that it is a valuable compromise. Since the models in POLAR are subject to change and replacement, it is important to have matched editions of the manual and the computer code. The edition date for this manual is September 1989, and it describes the POLAR 1.3 version delivered to AFGL in September 1989.

1.10 CODE STRUCTURE

POLAR is written in ASCII standard fortran using top down, structured programming principles and is in general accordance with Air Force coding standards. It consists of four main programs and several attending utility programs. Program one, called VEHICL, handles object definition. Program two, called ORIENT, is used to reorient an object and its grid. Program three, called NTERAK, actually calculates the spacecraft-plasma interaction and most of the physical models are contained in NTERAK. The fourth program is called SHONTL, and it controls plotting and information retrieval. The utility programs provide the means to translate machine independent output of SHONTL to local hardware graphics commands, perform quick one dimension calculations, and post-process NTERAK runs.

1.20 DOCUMENTATION

This document is structured hierarchically. The description of POLAR goes from basic physics, to physical models, to algorithms, to coding structure, and finally, to operating instructions.

2. THE PHYSICS OF LARGE STRUCTURES IN THE POLAR IONOSPHERE

A good source for information on the physics of large structures in the polar ionosphere is the POLAR Code Validation final report SSS-DFR-89-10708.

3. PHYSICAL MODELS EMPLOYED IN THE POLAR CODE

The POLAR code makes various assumptions which enable it to perform three-dimensional charge calculations in relatively short Debye length plasmas. In this section we examine the component physical models and discuss their validity. While each is addressed separately, the code achieves a self-consistent solution by various levels of iteration. These are described more fully in Chapter 4, Computational Techniques. This chapter provides an executive summary of the models as if the numerics were arbitrarily accurate. Numerical techniques are discussed in greater detail in Chapter 4.

One major, overriding assumption should be identified before the component by component description, which is that all time dependence on the scale of particle dynamics is ignored. This means that particles see spatially dependent but time independent fields for the period they are near the orbiting vehicle. As such, all plasma oscillations, including electron and ion modes are precluded. Thus, oscillations in the wake or at leading edges will not be predicted by the POLAR code.

3.10 THE POLAR PLASMA ENVIRONMENT

POLAR can model a wide variety of plasma environments from reasonable combinations of the following populations:

Ions:

Cool Maxwellian ions (input AMU).

Cool Maxwellian protons.

Both the protons and ions are assumed to be isotropic in the plasma frame. The relative densities are controlled by inputting the density ratio with the total constrained to equal the ambient electron density. Both populations have temperatures equal to the temperature of the

cool electrons, (temperature 1). During wake calculations the ion temperature can be defined to be different than the electron temperature.

Electrons:

Cool ambient Maxwellian, temperature 1, density 1.

Suprathermal, power law distribution of energies.

Hot Maxwellian, temperature 2, density 2.

Energetic, Gaussian distribution of energies.

The cool Maxwellian population is considered isotropic in the plasma frame. The other, more energetic populations may be given field-aligned and loss-cone distributions in the future but are presently considered isotropic only.

3.20 PLASMA POTENTIALS

The other major assumption is that the only fields of major importance are the static electric fields obtainable from Poisson's equation and the earth's magnetic field. The only velocity related field included is that induced by $\underline{v} \times \underline{B}$ on conducting surfaces. The frame of reference is chosen to be the stationary plasma, so that $\underline{v} \times \underline{B}$ effects appear on the vehicle as boundary conditions. The plasma at infinity is defined to be at zero potential.

Plasma potentials are obtained from Poisson's equation

$$\nabla^2 \phi = \lambda^{-2} \rho$$

where $\phi = eV/kT$ is the dimensionless potential, λ is the Debye length ($\lambda^2 = \epsilon_0 kT/Ne^2$), and ρ is the sum of the appropriate ion and electron charge densities ($\rho = N_i + n_e$). Contributions from hot auroral electrons and

particles backscattered from the vehicle are neglected except for electron secondaries generated during electron collection. Poisson's equation is solved using either fixed potential or fixed normal electric field boundary conditions on a surface by surface basis, as appropriate. Solution techniques for the Poisson equation are presented in Section 4.20, and the overall space charge iteration in Section 4.44.

3.30 PARTICLE DENSITIES

The electron density within ion collecting sheaths is assumed to be Maxwellian without any excluded orbits;

$$n_e = n_{e_0} e^{\phi/kT}$$

where n_{e_0} is the unperturbed cold component plasma density. The absence of excluded orbits implies neglecting potential barriers for electrons in the wake. The validity of this approximation has been studied using as an extreme case a disk moving infinitely fast with respect to thermal ions, but very slowly with respect to the electron thermal velocity. Such an object has rigorously no ion charge density in the wake and thus has the maximum negative space charge physically possible. Solving Poisson's equation for this case gives the maximum possible negative plasma potential. The central wake potential as a function of disk radius over Debye length is shown in Figure 3.30/1. We see that even for shuttle size objects that the peak space charge potential is less than 20 kT, or about 2 volts. If the surface boundary conditions are more negative than this wake space charge maximum, the potential in the wake will be monotonic and no electron orbits will be shadowed. This will clearly be the case for any case with substantial negative charging.

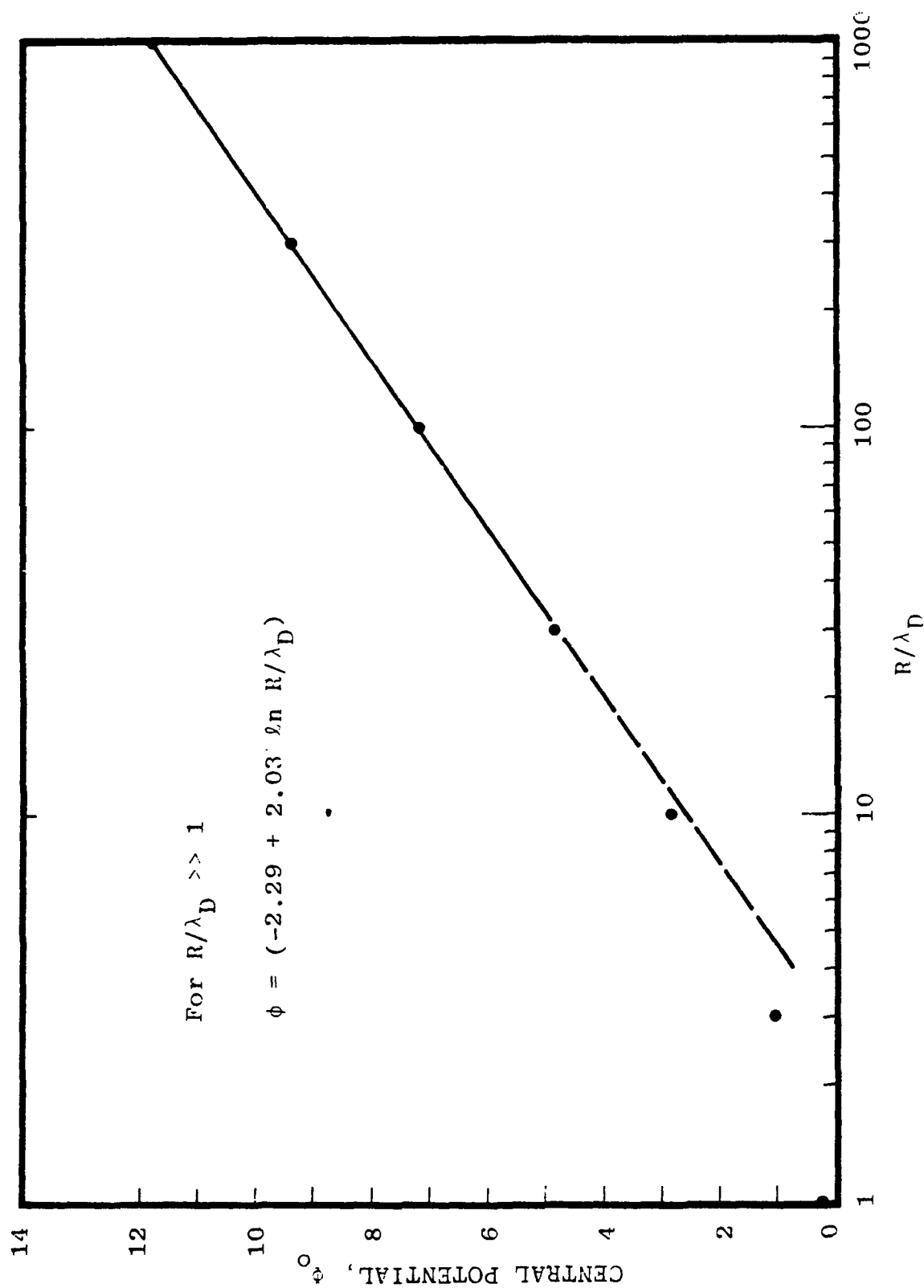


Figure 3.30/1. The central dimensionless potential ($\phi = eV/kT$) of a cylindrical ion void of radius R , with equilibrium filling of ambient Maxwellian electrons.

Within an electron sheath, the electron densities use a sheath electron model similar to the ion sheath model. The ion density term is determined using one of two models, depending upon the local potential. At large distances from the object, where the potential is near plasma ground (substantially less than the ram energy of the ions), ion orbits are assumed to be unperturbed by electric fields. In this presheath region, ion densities are determined for both ions and protons, by the "neutral ion model" described next. A sheath edge is assumed to separate the presheath from a sheath region wherein electric fields dominate thermal effects. In the sheath region, ion densities are determined by sheath ion model (3.32, 3.60).

It is important to note that the densities that result from the combined use of the neutral and sheath models are subject to certain limitations and shortcomings.

There is a low density or Laplace limit where the sheath edge is no longer sharply defined and electric fields extend far into the plasma. In this limit, the neutral ion approximation would fail, and thermal ion motion would be incorrectly ignored inside the sheath edge. This limit is characterized by a Debye screening length that is comparable to, or larger than, the object size. This does not mean POLAR cannot provide useful results in the long Debye length limit, since the space charge coupling to the potentials is reduced by λ^{-2} . The user should, however, understand that ion densities and sheath ion currents (3.60) can be in error. There is also a short Debye length limitation that occurs when the Debye length is very much less than a zone size and object potentials are low. This combination can result in an object to sheath-edge distance that should be less than a zone, which is, of course, impossible to model accurately. This limitation is further discussed in Section 4.44.

3.31 STRUCTURE OF THE PLASMA WAKE

The model of the wake structure used by POLAR depends on the position relative to the so-called ion front. This ion front marks the boundary where electron density begins to change on a scale commensurate with the Debye length and the ion density takes a sudden and dramatic drop. Several authors have discussed the relationship between the wake fill process and the theoretical problem of the expansion of a plasma into a vacuum. In particular, problems applicable to ionospheric conditions have been treated by Gurevich et al. (Ref. 3-8), Gurevich and Pitaevskii (Ref. 3-9), and Schunk (Ref. 3-10), to name a few.

The solution to the Vlassov-Poisson equation system is in general quite difficult to obtain, but for the expansion of a plasma into the void it can be solved explicitly [Gurevich et al., 1969] (Ref. 3-10). Ahead of the ion front the plasma is treated as rarefied: its motion is controlled by the thermal spread in ion velocities. Behind the front the motion is controlled by the electron temperature and ion mass. Figure 3.31/1 illustrates these regimes and defines the coordinate systems used.

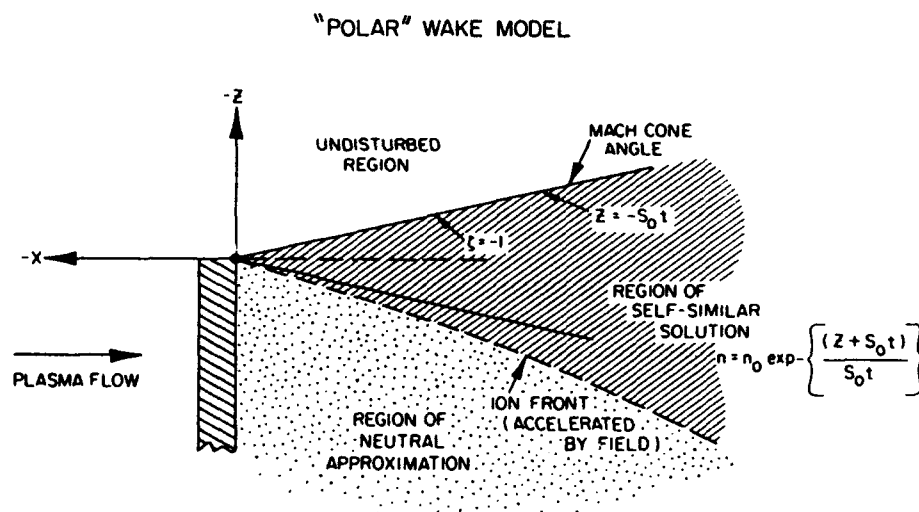


Figure 3.31/1. The POLAR wake code distinguishes three regions of interest. The ambient plasma, the region of self-similar model, and the neutral approximation spaces are bounded by the Mach cone $Z = -S_0 t$ and ion front, respectively. The coordinate system used is consistent with equations (1) through (10).

The governing equations in the region behind the front, considering that electrons are more mobile than ions and that they maintain equilibrium with a local potential, are

The Boltzman relation

$$n_e = n_0 \exp (e\phi/kT_e) \quad (1)$$

Continuity

$$\frac{\partial n_i}{\partial t} + \frac{\partial (n_i v)}{\partial z} = 0 \quad (2)$$

Equation of motion

$$\frac{\partial v}{\partial t} + \frac{v \partial v}{\partial z} = \frac{-e \partial \phi}{M \partial z} \quad (3)$$

Poissons equation

$$\frac{\partial^2 \phi}{\partial z^2} = 4\pi e (n_e - n_i) \quad (4)$$

where

- n_0 ambient density:
- n_i ion density:
- n_e electron density:
- T_e electron temperature:
- e electron charge:
- ϕ local potential:
- k Boltzman's constant.

and where z is a variable representing distance parallel to the front velocity or, in this case, perpendicular to the orbital velocity.

Crow et al. [1975] (Ref. 3-12) have numerically solved (1) through (4) to predict the position of the ion front. Katz et al. [1985] (Ref. 3-13) developed an analytical fit to the Crow results:

$$Z_F(t) = 2\lambda_d \left\{ \left(\omega t + \frac{1}{a} \right) \ln(1 + a\omega t) - \omega t - \left(1 - \frac{0.429}{a} \right) \left(\omega t - \frac{1}{a} \ln(1 + a\omega t) \right) \right\} \quad (5)$$

where

$$\omega = \frac{(4\pi n_0 e^2)^{1/2}}{M} \quad \lambda_d = \frac{(kT_e)^{1/2}}{4\pi n_0 e^2}$$

are the ion plasma frequency and Debye length, respectively, and a is a free parameter determined to be ≈ 1.6 .

Katz et al. [1985] showed that this formula agrees well with laboratory data from Wright et al. [1985] (Ref. 3-14) and incorporated it in POLAR. Ahead of this front Z_F , the plasma is assumed to expand owing to thermal motion, the so-called "neutral approximation." Behind Z_F the plasma evolves into a state which is self-similar [Chan et al., 1984 (Ref. 3-15)]. The self-similar solution of (1)-(4) for $z > -S_0 t$ is

$$n = n_0 \exp \left[- \frac{(z + S_0 t)}{S_0 t} \right] \quad (6)$$

where $S_0 = (kT_e/M)^{1/2}$ is the ion acoustic speed.

The time variable is defined as

$$t = \frac{x}{V_0} \quad (7)$$

where x is the distance behind the object (perpendicular to z) and V_0 is the orbital velocity. We define the self-similar variable ξ as

$$\xi = \frac{z}{S_0 t} \quad (8)$$

Thus the self-similar solution essentially states that between the region bounded in positive z by the front Z_F and in negative z by the line $z = -S_0 t$, the density rises exponentially to be equal to the ambient value along $z = -S_0 t$. This is an intuitively reasonable result.

In summary, the wake routines in POLAR employ two limiting cases.

(1) Ahead of the ion front the electric field is negligible and the motion of ions is identical to neutrals. (2) Behind the ion front, whose position is determined by (5), the quasi-neutral self-similar solution of (6) is implemented.

POLAR has routines which model accurately the geometry of the object, and the "neutral ion" trajectories are calculated from

$$f_i(x, v) = g(x, \Omega) f_{i0}(v) \quad (9)$$

where $f_{i0}(v)$ is the unperturbed distribution function for a drifting Maxwellian, and $g(x, \Omega)$ has value "0" if a ray starting from x and going in the direction Ω would strike the vehicle and "1" if it would not.

The local density is given by

$$n_i(x) = \int f_i(x, v) = \int g(x, \Omega) \left\{ \int f_{i0}(v, \Omega) v^2 dv \right\} d\Omega \quad (10)$$

This initial density calculated in three dimensions for neutral particles is compared with density calculated assuming the complex geometric object is replaced by a flat plate at a position where the dominant source appears at the object edge. This ratio provides a "geometric correction factor," which is applied to the quasi-neutral one-dimensional solution discussed earlier for positions behind Z_F . In this way, POLAR can calculate quite rapidly an approximate value for the ion and electron densities in the wakes of complex objects.

Note that the assumptions behind the front are (1) that the electron temperature and ion mass govern the equation of motion, (2) that the plasma is quasi-neutral, (3) that the magnetic field does not

affect the ion or electron motion, (4) that equation (5) serves as a good approximation for determining the boundary of the ion front, and (5) that the geometric correction factor calculated in detail with the three-dimensional neutral model can be approximately applied to correct the plasma densities as well. Therefore the algorithm can address complex geometries but takes advantage of the smooth wake structure characteristic of ionospheric plasmas where $T_i/T_e \approx 1$. Additionally, the model ignores fields existing in a sheath near the body surface, which should not be of concern in cases where the spacecraft is near plasma potential. This implies that ion acceleration calculate by POLAR is dominated by electric fields due to space charge separation in the wake. (Ref. 3-16)

3.32 SHEATH DENSITIES

Section 3.60 discusses the POLAR sheath model and the sharp edged sheath concept. The derivation of densities from trajectories within the sheath is explained in 4.42.5. What follows here is a brief outline.

Given a sheath edge, currents from "infinity" to the sheath edge are calculated analytically using orbit-limited theory. These currents are assigned to a set of "super particles" (Reference 3-5) that are tracked inwards from the sheath edge to the vehicle surface. Pushed particle densities are determined from the product of the particle current and the time that a particle spends in an element.

3.40 SURFACE CURRENTS

POLAR models a number of charged particle sources as responsible for surface and vehicle charging. These are ambient ions and electrons, energetic electrons, ion and electron generated secondary electrons, backscatter electrons, photoelectrons, and particle beams.

The comments of Section 3.30 concerning potential barriers apply here when the vehicle is negatively charging. We assume the ambient electrons to be repelled with no excluded orbits within the hemisphere of velocities impinging upon a surface. When these conditions are met, the velocity space integrals over a Maxwellian distribution decouple from the surface potential and we may write

$$j_e(V) = j_{eo} e^{eV/kT}$$

For the energetic electron sources, the current integrals are more involved. These are discussed further in Section 3.41 and presented in Sections 4.52.3 - 4.52.5.

The calculation of attracted specie currents is discussed in Section 3.42.

3.41 ANALYTICAL ELECTRON SURFACE CURRENTS

A statistical study (Ref. 3-1) of high latitude precipitating electrons has shown that these fluxes can be well represented by the following parametric expression

$$\Phi(E) = AE^{-\alpha} + Cn \frac{E}{(kT)^{3/2}} e^{-E/kT} + E B e^{-[(E-E_0)/\delta]^2} \quad (3.41-1)$$

These are the power law, hot Maxwellian and Gaussian distributions mentioned in Section 3.10, where $C = (2 m_e)^{-1/2} \cdot \pi^{-3/2}$, and A , α , n , T , B , E_0 and δ are parameters determined by the particular shape of a spectrum. Here, $\Phi(E)$ has units of $\#/\text{m}^2 \cdot \text{s} \cdot \text{str} \cdot \text{keV}$. To apply these distributions to the charging of a surface, it is necessary to formulate the distribution function, f , at the surface. We start by dissecting Eqs. (3.41-1). Equation (3.41-1) appears to assume a zero space potential because a factor of E (total energy) rather than K (kinetic energy) is used for the velocity or energy space differential volume unit. We next invoke the Vlasov equation to allow a mapping of f along a trajectory connecting the surface to "infinity". In doing so, we replace the one factor of E with K , while elsewhere setting $E = K + qV$ where q is the particle charge, and V is the surface voltage. Thus, we write,

$$\begin{aligned} f(K, V, \psi) = & A(\psi) * (K + qV)^{-(\alpha+1)} \\ & + \sum_{e=1}^2 \frac{F_e(\psi)}{(kT_e)^2 \pi} * \exp(-(qV + K)/k T_e) \\ & + B(\psi) \exp(-(K - K_0)^2/\Delta^2) \end{aligned}$$

where the index e covers both hot and cold electrons and

where the index e covers both hot and cold electrons and

$$F_e(\psi) = g(\psi) * n_e * \sqrt{kT/2\pi m}$$

is the thermal flux multiplied by a function of the pitch angle. The net electron current density at a surface is, of course,

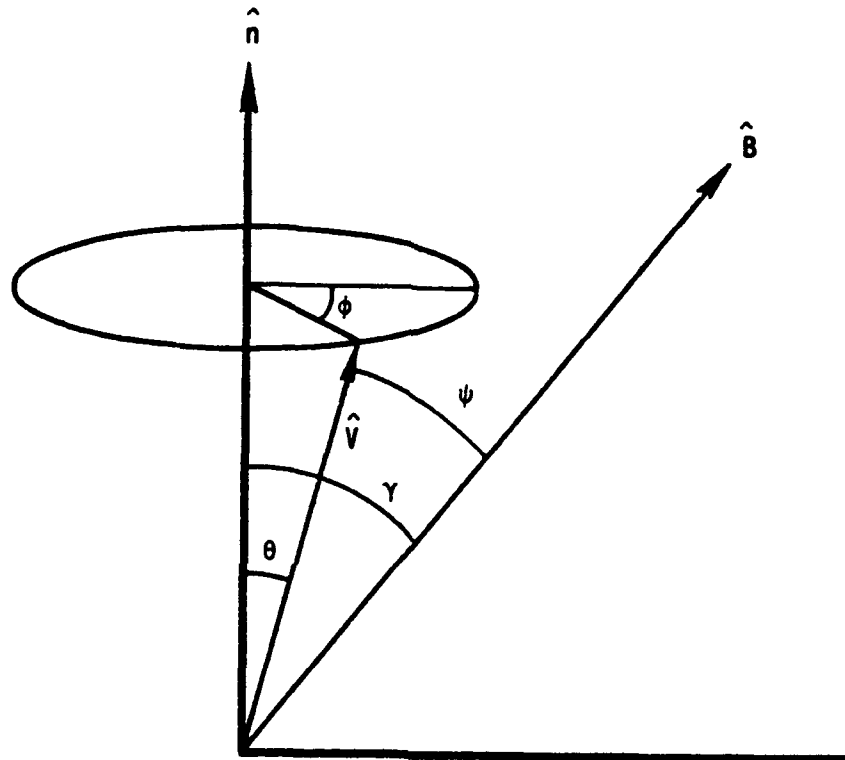
$$J = q \int_0^{\pi/2} d\phi \int_0^{2\pi} d\theta \int_L^U k dk f(k, V, \psi) \cos\theta \sin\theta .$$

$$L = \max(0, -qV)$$

The relation between the pitch angle ψ , and the spherical polar angles of the surface normal is:

$$\psi(\theta, \gamma, \phi) = \cos^{-1} [\cos\gamma \cos\phi + \sin\gamma \sin\theta \cos\phi]$$

where γ is the angle between the surface normal and the magnetic field. These angles are illustrated in Figure 3.41/1.



$$\psi(\theta, \gamma, \phi) = \cos^{-1} [\cos\gamma \cos\theta + \sin\gamma \sin\theta \cos\phi]$$

Figure 3.41/1.

It is important to note that in composing f at a surface from f at infinity, the angular factor in f , $A(\psi)$, $g(\psi)$, and $B(\psi)$, may evolve dramatically. Ultimately, POLAR may estimate the angular evolution of the electron distribution function, as this may be important for some narrow high energy distributions as well as necessary for the prediction of magnetic field effects. However, since the usual tendency is for the repelled specie distribution to broaden our first guess will be to assume f to be isotropic at all surfaces.

The energy integration limits, U and L , are 0 and ∞ for the Gaussian and Maxwellian distributions, but a lower cutoff must be imposed on the power law distribution. This cutoff is physical in its origin as the electron currents are finite, but determining it accurately is not always possible. POLAR uses a 100 eV default cutoff which may be changed, or alternatively the total current for these electrons may be specified and a reasonable cutoff will be deduced by POLAR.

POLAR integrates each population separately. These integrations are described in Section 4.52.

3.42 ATTRACTED PARTICLE SURFACE CURRENTS

POLAR presently considers two positively charged particle sources; one specie of ion (singly charged with a variable mass), and protons. As the attracted species, the calculation of the ion current density at a surface is a non-local problem which often depends critically upon the shape of the orbits that bring ions to the surface (Ref. 3-2). That is, the problem is generally numerical with few exceptions that yield to analytic evaluation. POLAR calculates these currents as part of its sheath model, so the reader is referred to Section 3.60 for further information, and a brief discussion follows here.

External to a sheath edge "orbit-limited" (Ref. 3-2 and Section 3.60) conditions may be assumed which will allow a flowing Maxwellian velocity distribution to be analytically integrated to find the ion currents to the sheath edge. These currents are assigned to representative particles that are traced inwards to the object surface to yield ion surface currents. This calculation is performed as a portion of the POLAR sheath model which is executed as the CURREN module of NTERAK (see Chapter 5 for POLAR code structure).

When electrons are the attracted specie, the same procedure as above is followed, except that external to the sheath the electrons are assumed to be thermally distributed and at the sheath edge the one-sided thermal flux through the boundary is used to define the electron sheath flux.

In addition to the space charge limited surface current model, two other methods can be used to calculate the attracted specie current.

The first method applies the analytical expression for orbited limited current to a spherical probe, using the local surface potential for the sphere potential,

$$j = j_{th} \times \left(1 + \frac{|\phi|}{\phi}\right)$$

for the attracted species. The effective temperature, θ , takes into account the orbital velocity for ions. This is the same approximation used in the NASCAP/GEO spacecraft charging code.

The second method uses the analytical convergence formula to obtain a potential and a sheath boundary surface. The total sheath current is distributed among the surface cells according to their surface potentials, just as in the orbit limited case. The only difference is the currents are normalized so their sum equals the total sheath current.

3.43 INITIAL SHEATH PARTICLE VELOCITY DISTRIBUTION

In this section we discuss the distribution of the initial particle velocities to be tracked from the sheath. Much of this is based on the results of Section 4.42.2 which discusses the effect of attractive potentials on a flowing plasma. In that section the one component of the first moment was computed, namely,

$$\langle \vec{v} \cdot \vec{n} \rangle$$

Where $\langle \rangle$ refers to averaging of the distribution function at a surface whose normal is \vec{n} (see figure 4.42/1).

The distribution of initial velocities is selected to produce the same mean and variance of the actual distribution function. This is to simulate the thermal spread of the initial velocities and produce orbit limited effects. That is

$$\vec{V} = \langle \vec{v} \rangle \text{ and,}$$

$$\sigma_{ij} = \langle (v_i - V_i)(v_j - V_j) \rangle$$

Where the subscripts refer to the directions perpendicular to the surface normal. For convenience we choose $i=1$ to be in the plane defined by the surface normal and the mach vector, and $i=2$ to be in the direction normal to both the surface normal and the $i=1$ direction. For a cold plasma, $t=0$, $\sigma_{ij} = 0$ and the initial velocities have no thermal spread. The results of section 4.42.2 were extended to allow the calculation of the σ_{ij} . The initial particle distribution consists of 5 particles,

$$\{\vec{u}_1, \vec{u}_2, \vec{u}_3, \vec{u}_4, \vec{u}_5\},$$

where

$$\vec{u}_3 = \vec{V}, \text{ and}$$

$$\vec{u}_{1,2} = \vec{V} \pm \sigma_{11}$$

$$\vec{u}_{3,4} = \vec{V} \pm \sigma_{22}$$

This distribution reproduces the mean and variance of the original distribution.

3.50 ELECTRICAL CHARGING

The electrical charging of the spacecraft is modeled using a circuit analogy. The plasma around the craft becomes a current source with a capacitance between the plasma and the object's surface. The spacecraft is modeled as a network of capacitors, resistors, and voltage sources. The basic charging equation is

$$\tilde{I}(t) = \tilde{C} \frac{d}{dt} \tilde{V}(t) + \tilde{\sigma} \tilde{V}(t) \quad (3.50-1)$$

where I is current, C capacitance, σ conductance, and V is voltage. Each surface (the smallest mesh unit sized square, triangular and rectangular building block) contributes a component to the current and voltage vectors.

Surface voltages are updated (integrated) by timestepping a finite difference approximation to Eq. (3.50-1) (Section 4.51). This integration frequently proves to be difficult because of the wide range of capacitance that can occur in the \tilde{C} matrix. For instance, a surface to plasma capacitance might be

$$\frac{C_{sp}}{A} \approx \frac{\epsilon_0}{R} \approx 10 \text{ pf/m}^2$$

where A is the surface area and R is an effective radius, whereas the surface to conductor capacitance of a dielectric might be

$$\frac{C_b}{A} \approx \frac{\epsilon}{d} \approx 0.1 \text{ } \mu\text{f/m}^2 .$$

Thus, a driving current of 10^{-5} A/m^2 would produce charging rates of 10^6 volts/sec and 10^2 volts/sec. Obviously, these two extremes would require different timesteps for a simple explicit integration.

The stability difference between the explicit and implicit forms can be demonstrated by a simple scalar analog.

$$\text{Explicit: } C[V(t_2) - V(t_1)] = I(t_1) \cdot \Delta t$$

$$\text{Implicit: } C[V(t_2) - V(t_1)] = I(t_2) \cdot \Delta t$$

Substituting

$$I(t_2) = I(t_1) + \left. \frac{dI}{dV} \right|_1 \quad (3.50-2)$$

gives

$$V(t_2) - V(t_1) = \frac{I(t_1) \Delta t}{C - \left. \frac{dI}{dV} \right|_1}$$

If we take a case of $C = 10^{-11} \text{ f}$, $I(t_1) = 10^{-6} \text{ amp}$, $\Delta t = 1 \text{ sec}$, $dI/dV = -10^{-8} \text{ amp/volt}$ we find:

$$\text{Explicit: } \Delta V = 10^5 \text{ volts.}$$

$$\text{Implicit: } \Delta V = 10^2 \text{ volts.}$$

That the explicit answer is unstable is indicated by plugging ΔV into Eq. (3.50-2) giving $I(t_2) = -10^{-3} \text{ amp}$ (explicit) or $I(t_2) = 10^{-9} \text{ amp}$ (implicit).

POLAR utilizes a two stage implicit timestepping algorithm to allow large timesteps for the large capacitances while maintaining accuracy and stability for the smaller capacitances. Details of implementation can be found in Sections 4.51 and 5.70.

3.60 THE POLAR SHEATH MODEL

The concept of a plasma sheath requires definition. In general, the plasma sheath can be defined to be the region of non-neutral charge density that shields a charged body from distant plasma. A more precise definition should distinguish between an "orbit-limited" sheath and a "space charge-limited" sheath. Investigations into current collection by Langmuir probes (Ref. 3-2, 3-3) in the long Debye length limit, have shown that current collection is orbit-limited, i.e., on the surface of a probe, distribution functions are filled over a hemisphere, and are related to the distant plasma distribution function by constants of the motion or "orbits". As the Debye length is shortened, current collection remains orbit-limited until the charge density is sufficient to cause the electric potential to decrease faster than the inverse square of the radial distance. At this point, current collection is said to be space charge-limited.

An important feature of the orbit-limited sheath is that the particle currents to a surface are independent of the exact shape of the potential well, making it possible to derive general expressions for currents and densities. The opposite is true of the space charge-limited sheath where currents and densities must be calculated numerically by following trajectories in potential wells that must be consistent with the particle densities. The many approaches to this problem have been reviewed recently by Laframboise (Ref. 3-4).

POLAR models the space charge-limited extreme, which dictates the use of some trajectory tracing. Efficiency is maintained by recognizing that orbit-limited conditions exist in the quasi-neutral region outside the sheath, and that trajectories must be followed only inside the sheath. POLAR thus makes a sharp sheath edge approximation to divide a problem into the two regimes. Fluxes from "infinity" to the sheath edge are calculated analytically using orbit-limited theory (Sections 3.42, 4.42.2). These fluxes are then assigned

(Section 4.42.3) to trajectories that are tracked (Section 4.42.4) through the sheath to determine particle densities in the sheath (Section 4.42.5) and surface currents (Section 4.53). Of course, for the self-consistent probe problem, POLAR must iterate between sheath density solutions and Poisson solutions, and for a charging problem a higher level of iteration updates the surface potentials and iterates with the sheath-Poisson solution.

The sheath edge is nominally chosen to be the -0.47 kT/e potential contour for ions and $+0.47 \text{ kT/e}$ for electrons. For a spherical probe in a non-flowing plasma, this is consistent with previous investigations (Refs. 3-5, 3-6). In the presence of net plasma flow, POLAR maintains the -0.47 kV sheath edge choice, but it is presently not clear what the best choice for the sheath edge will be for high flow problems (see Section 4.42 for details on edge definition).

Numerical considerations can also mediate the choice of the sheath edge. In the limit of high density and short Debye length (with respect to mesh spacing), a necessary stabilization procedure for the Poisson solver (Section 4.44.1) will cause an expansion of the sheath. POLAR compensates for this by choosing the sheath edge at a slightly higher potential (Section 4.44.2), while defining a "presheath edge" potential that is still -0.47 kT/e . This presheath edge potential is used to calculate the orbit limited fluxes to the sheath. This technique compensates for the sheath expansion by reducing the expanded sheath area, while keeping the input fluxes constant.

There are two major approximations made in the POLAR sheath model. The first is the so-called sharp sheath edge approximation. This assumes that there is a sharp boundary between non-neutral sheath and the surrounding quasi-neutral presheath region. For large objects in short Debye length plasmas this is a very good approximation. The other

approximation is that thermal effects within the sheath are small, i.e., from a given position on the sheath boundary, a single trajectory is adequate to represent all particles entering from that position. This implies that potentials, V , exist in the sheath such that

$$\frac{eV}{kT} \gg 1$$

and that the electric fields near the sheath edge are sufficiently strong so that a velocity space element is accelerated rapidly and its thermal spread is small;

$$v_{th} \cdot \tau \ll L$$

where v_{th} is the ion thermal velocity, τ is the transit time from the sheath edge to the vehicle, of characteristic dimension L .

REFERENCES, CHAPTER 3

- 3-1 Fontheim, E. G., K. Stasiewicz, M. O. Chandler, R.S.B. Ong and E. Gombosi, "Statistical Study of Precipitating Electrons," J. Geophys. Res., Vol. 87 (A5), 1982. pp. 3469-3480.
- 3-2 Laframboise, J. G., "Theory of Spherical and Cylindrical Langmuir Probes in a Collisionless, Maxwellian Plasma at Rest," UTIAS Report No. 100, 1966.
- 3-3 Laframboise, J. G. and L. W. Parker, "Probe Design for Orbit-Limited Current Collection," Phys. of Fluids, Vol. 16, N5, 1973.
- 3-4 Laframboise, J. G., "Is There a Good Way to Model Spacecraft Charging in the Presence of Space-Charge Coupling, Flow, and Magnetic Fields," in Proceedings of the Air Force Geophysics Laboratory Workshop on Natural Charging of Large Space Structures in Near Earth Polar Orbit, AFGL-TR-83-0046, September 1982, ADA134894.
- 3-5 Parker, L. W., "Computations of Collisionless Flow Past a Charged Disk," NASA CR-144159, 1976.
- 3-6 Parrot, M.J.M., L.R.O. Storey, L. W. Parker and J. G. Laframboise, "Theory of Cylindrical and Spherical Langmuir Probes in the Limit of Vanishing Debye Number," Phys. Fluids, Vol. 25(12), December 1982.

- 3-7 Gurevich, A. V. and L. P. Pitayevsky, "Hypersonic Body Motion Through Rarefied Plasma," *Physical Review Letters*, Vol. 15, 8, August 23, 1965, pp. 346-348.
- 3-8 Gurevich, A. V., L. P. Pariiskaya and L. P. Pitaevskii, "Self-similar Motion of a Rarefied Plasma," *Sov. Phys. JETP (Eng. Transl.)* Vol. 22, 1966, p. 449.
- 3-9 Gurevich, A. V. and L. P. Pitaevskii, "Non-linear Dynamics of a Rarefied Ionized Gas," *Prog. Aerospace Sci.*, Vol. 16, 1975, p. 227.
- 3-10 Singh, N. and R. W. Schunk, "Numerical Calculations Relevant to the Initial Expansion of the Polar Wind," *J. Geophys. Res.*, Vol. 87, 1982, p. 9154.
- 3-11 Gurevich, A. V., L. P. Pitaevskii and V. V. Smirnova, "Ionospheric Aerodynamics," *Space Sci. Rev.*, Vol. 9, 1969, p. 805.
- 3-12 Crow, J. E., P. J. Aver and J. E. Allen, "The Expansion of a Plasma into a Vacuum," *J. Plasma Phys.*, Vol. 14, 1975, p. 65.
- 3-13 Katz, I., D. E. Parks and K. H. Wright, Jr., "A Model of the Plasma Wake Generated by a Large Object," *IEEE Transl. Nucl. Sci.*, Vol. NS-32(6), 1985, p. 4092.
- 3-14 Wright, K. H., Jr., N. H. Stone and U. Samir, "A Study of Plasma Expansion Phenomena in Laboratory-generated Plasma Wakes: Preliminary Results," *J. Plasma Phys.*, Vol. 33, 1985, p. 71.
- 3-15 Chan, C., N. Hershkowitz, A. Ferreria, T. Intrator, B. Nelson and K. Lonngren, "Experimental Observations at Self-similar Plasma Expansion," *Phys. Fluids*, Vol. 27, 1984, p. 266.
- 3-16 Murphy, G. and I. Katz, "The POLAR Code Wake Model: Comparison with In Situ Observations," *J. Geophys. Res.*, Vol. 94(A7), 1989, pp. 9065-9070.

4. COMPUTATIONAL TECHNIQUES

This section describes the algorithms used to implement the physical model.

4.10 GRIDS - DISCRETIZATION OF SPACE

POLAR is a three-dimensional computer code; that is, its internal representation of space allows variations in all three coordinate directions. Since problem set-up can be extremely complex in three-dimensions, the choice was made to keep the spatial coordinate system as simple as possible. Space is divided uniformly into cubes. The computer code stores information about a large set of cubical volumes, called elements. It also stores values of electric potentials for the corners of each element. The corners are referred to as nodes.

4.11 STAGGERED MESH

The coordinate system used in POLAR is Cartesian. This greatly simplifies object definition and converting position vectors into element locations. However, the types of problems POLAR is designed to handle are very anisotropic. The density and potential perturbations are along the wake direction and can extend for several vehicle diameters. To provide resolution in this wake region but not occupy an excessive amount of computer storage, a system of staggered meshes has been implemented. The staggered mesh consists of rectangular layers of elements that are only one mesh unit deep in the z-direction. They are stacked upon each other so that the center of each layer is as close to the wake center as possible and still have the nodes at integer coordinates. Since this extension always is in the z-direction, the object coordinates can be transformed by a 90° rotation matrix so that an arbitrary Mach vector can be accommodated.

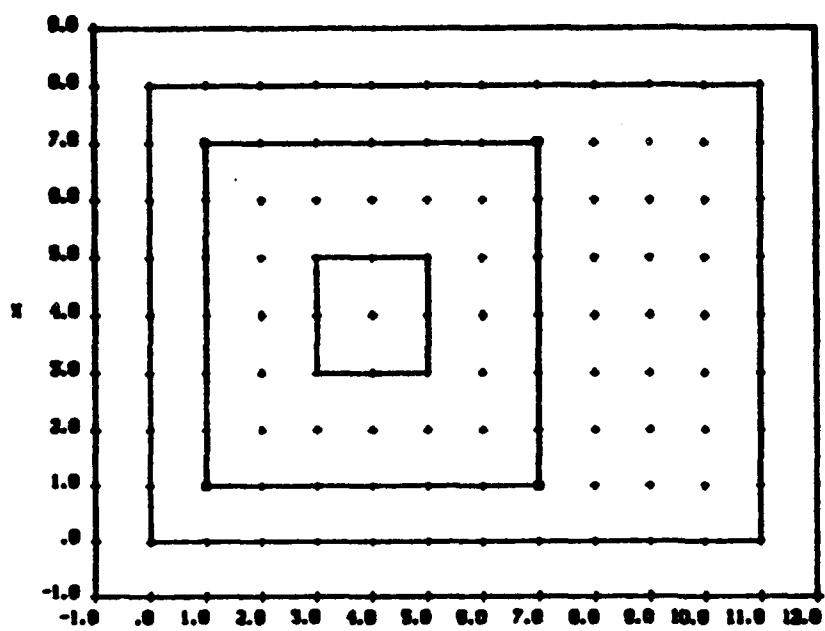
The problem space is largest in the z-direction so that viable results may be obtained for large Mach vectors in lower density plasma.

Examples of grids for two objects are shown in Figure 4.11/1.

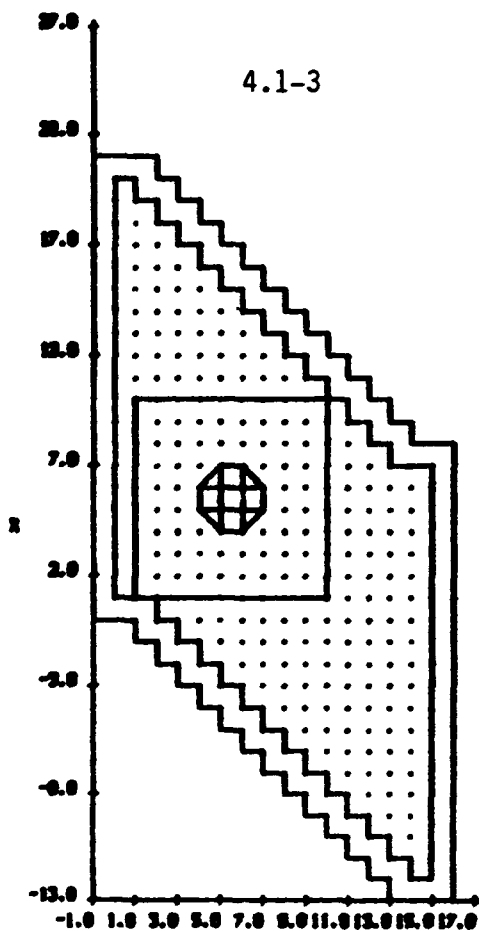
4.12 OBJECT GRID

The object grid is the space in which the object is defined. This subsection of the grid space is non-stepped and is a rectangular prism. The object must be defined so that it fits entirely within this space without touching the grid boundary.

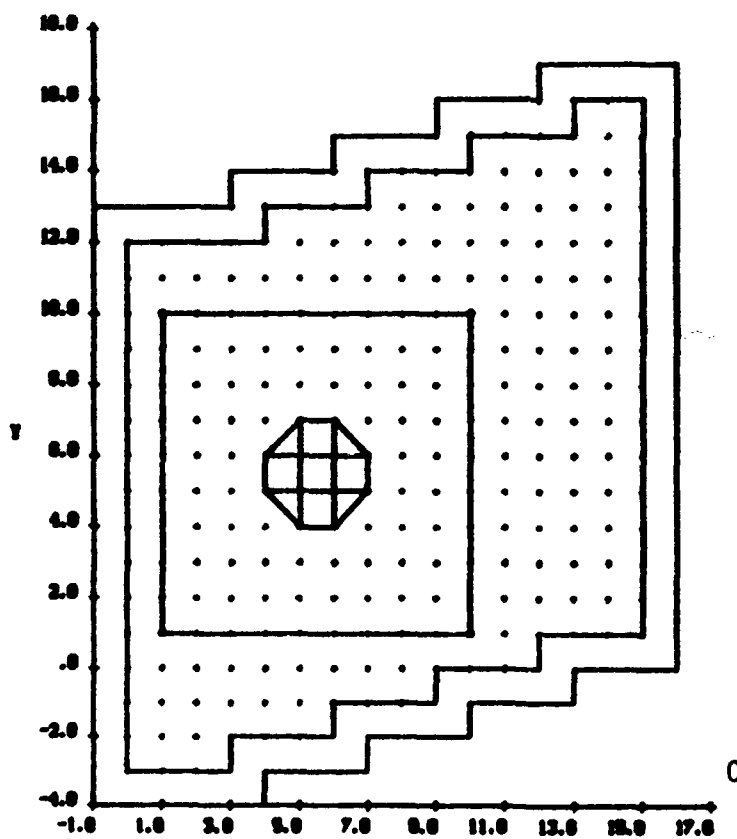
The object grid also provides the coordinate reference point for the entire problem. The lowest point on the x, y and z axis (lowest leftmost corner) is defined to be the origin and has the coordinate value of (1,1,1).



A



B



C

Figures 4.11/1. (A) Cube moving to left. (B) Quasisphere moving to left and rising. (C) Quasisphere moving to left and falling.

4.20 POTENTIAL CALCULATIONS

4.21 FINITE ELEMENTS

4.21.1 General

Consider a charged object isolated in space. The potential ϕ everywhere is given by the solution to Poisson's equation

$$-\epsilon \nabla^2 \phi = \rho \quad (4.1)$$

Since POLAR considers some portion of the charge density ρ to be dependent on the local potential, we can separate the local and non-local contributions to ρ , and linearize about some estimate of the local potential, ϕ_0 ,

$$-\epsilon \nabla^2 \phi = \rho_{ne} + \rho_e(\phi) + \rho'_e(\phi - \phi_0) = \rho_0 + \rho' \phi$$

where

$$\rho' = \partial \rho / \partial \phi .$$

The variational principal associated with this equation is

$$\frac{\partial}{\partial \phi} \left[\int dv \left(\frac{1}{2} (\nabla \phi)^2 - \frac{\rho_0 \phi}{\epsilon} - \frac{\rho' \phi^2}{2\epsilon} \right) - \int_{c_S} \frac{\sigma \phi}{\epsilon} dS - \int_{c_B} \phi \cdot \phi \cdot dS' \right] = \frac{\delta}{\delta \phi} \quad L = 0$$

where we integrate over both the object and boundary surfaces (c_S , c_B).

To simplify things for the purpose of illustration, let us fix the potentials on these surfaces. L then simplifies to

$$L = \int dV \left[\frac{1}{2} (\nabla\phi)^2 - \frac{\rho_o\phi}{\epsilon} - \frac{\rho'\phi^2}{2\epsilon} \right] \quad (4.2)$$

Equation (4.2) involves an integral over the volume of the computational space. One way to treat this integral is to divide the space up into finite cubic volume elements. We begin with the first term in Eq. (4.2).

$$\int dV \frac{1}{2} (\nabla\phi)^2 = \sum_e \int_{V_e} dV_e \frac{1}{2} (\nabla\phi)^2$$

In this approach the potential ϕ is defined at each grid point, or node, defining the vertices of the elements. The potential inside each element is then trilinearly interpolated from the values of each of its eight vertices.

$$\phi^e(x,y,z) = \sum_{i \in e} N_i^{xyz} \phi_i$$

where " i " are the nodes of element " e ", and the N_i are given in Section 4.21.21. We form now,

$$\nabla\phi^e(x,y,z) = \sum_i \nabla N_i^{xyz} \phi_i$$

and

$$\begin{aligned} \int dV \frac{1}{2} (\nabla \phi)^2 &= \frac{1}{2} \sum_{\mathbf{e}} \int dV_{\mathbf{e}} \sum_i \sum_j \nabla N_i^{\mathbf{e}} \nabla N_j^{\mathbf{e}} \phi_i \phi_j \\ &= \frac{1}{2} \sum_{\mathbf{e}} \sum_i \sum_j W_{ij}^{\mathbf{e}} \phi_i \phi_j \end{aligned}$$

where we have defined the quantity,

$$W_{ij}^{\mathbf{e}} = \int_{\mathbf{e}} dV_{\mathbf{e}} \nabla N_i^{\mathbf{e}} \cdot \nabla N_j^{\mathbf{e}} \quad (4.3)$$

Note that W_{ij} depends only upon the shape of the element " \mathbf{e} " (i.e., whether the cube is empty or partially filled). Similarly, for constant charge in an element, the second term in Eq. (4.2) becomes

$$\begin{aligned} \int dV \frac{\rho_o \phi}{\epsilon} &= \frac{\rho_o}{\epsilon} \sum_{\mathbf{e}} \sum_i \phi_i \int_{\mathbf{e}} N_i dV_{\mathbf{e}} = \frac{\rho_o}{\epsilon} \sum_{\mathbf{e}} \sum_i R_i^{\mathbf{e}} \phi_i \\ R_i^{\mathbf{e}} &= \int_{\mathbf{e}} N_i dV_{\mathbf{e}} \end{aligned}$$

Treating the last term,

$$\begin{aligned} \int \frac{\rho' \phi^2}{2\epsilon} dV &= \frac{\rho'}{2\epsilon} \sum_{\mathbf{e}} \sum_i \sum_j \phi_i \phi_j \int_{\mathbf{e}} N_i N_j dV_{\mathbf{e}} = \frac{\rho'}{2\epsilon} \sum_{\mathbf{e}} \sum_i \sum_j V_{ij}^{\mathbf{e}} \phi_i \phi_j \\ V_{ij}^{\mathbf{e}} &= \int_{\mathbf{e}} N_i N_j dV_{\mathbf{e}} \quad (4.4) \end{aligned}$$

The variational principle therefore becomes:

$$\frac{\delta}{\delta \phi_i} \left[\sum_e \sum_i \left[\sum_j \frac{1}{2} (W_{ij}^e - \frac{\rho'}{\epsilon} V_{ij}^e) \phi_i \phi_j - \frac{\rho_o}{\epsilon} R_i^e \phi_i \right] \right] = 0 \quad (4.5)$$

$$\sum_e \sum_j \left[(W_{ij}^e - \frac{\rho'}{\epsilon} V_{ij}^e) \phi_j - \frac{\rho_o}{\epsilon} R_i^e \right] = 0 \quad \text{for each } i \quad (4.6)$$

is the equation that POLAR solves element by element, under the condition of fixed boundary and object potentials. POLAR (or more correctly NTERAK) solves Eq. (4.6) using the conjugate gradient method that is presented in Section 4.31.

4.21.2 SPECIAL CELLS

The geometry of POLAR objects can be treated in terms of a relatively small number of volume cells. Each type has a maximum of eight corner nodes, plus a node at the center of each possible surface pointing into the element. The most common volume element (designated type 0) is the empty cube with no surfaces. Other elements are:

- a. The empty cube with up to 6 surfaces (also type 0);
- b. The wedge element (type 1);
- c. The empty element with a diagonal line (produced by a right-triangle surface or a slanted thin plate) on one face (type 2);
- d. The tetrahedron (type 3);
- e. The truncated cube (type 4);
- f. The slanted thin plate (type 5).

Each element is characterized by (1) a standard orientation; (2) a set of interpolation functions (see 4.21.21 for treatment of surface nodes); (3) the matrix \underline{W} , given by Eq. 4.3, which represents the operator $-\nabla^2$ in Poisson's equation; and (4) the matrix \underline{V} , defined by Eq. (4.4), which handles the screening part of Poisson's equation.

4.21.21 INTERPOLATION FUNCTIONS FOR FACE-CENTERED SURFACE NODES (FCSN'S)

In constructing the matrices $\underline{\underline{W}}$ and $\underline{\underline{V}}$ for volume cells with FCSN's it is convenient to work with the vector $\underline{\underline{\psi}} = \underline{\underline{T}}^T \underline{\underline{\phi}}$, where, for corner nodes, $\underline{\underline{\psi}}$ has identical entries to the potential vector $\underline{\underline{\phi}}$, but for an FCSN the $\underline{\underline{\psi}}$ entry is the difference between its electrostatic potential and the average of that of its corners. In terms of $\underline{\underline{\psi}}$, corner node interpolation functions are constructed neglecting the FCSN's, and FCSN interpolation functions are unity at the FCSN and zero on all other faces. In terms of these interpolation functions, the matrices $\underline{\underline{W}}$ and $\underline{\underline{V}}$ are defined by

$$\int |\underline{\underline{\nabla}} \underline{\underline{\phi}}|^2 d^3 \underline{\underline{r}} = \underline{\underline{\phi}}^T \underline{\underline{W}} \underline{\underline{\phi}} = \underline{\underline{\psi}}^T \underline{\underline{A}} \underline{\underline{\psi}}$$

$$\int \underline{\underline{\phi}}^2 d^3 \underline{\underline{r}} = \underline{\underline{\phi}}^T \underline{\underline{V}} \underline{\underline{\phi}} = \underline{\underline{\psi}}^T \underline{\underline{B}} \underline{\underline{\psi}}$$

where the integrals are over the element volume, and we have defined $\underline{\underline{A}}$ and $\underline{\underline{B}}$, which are readily shown to be given by

$$A_{ij} = \int \underline{\underline{\nabla}} N^i \cdot \underline{\underline{\nabla}} N^j d^3 \underline{\underline{r}}$$

$$B_{ij} = \int N^i N^j d^3 \underline{\underline{r}} .$$

Finally,

$$\underline{\underline{W}} = \underline{\underline{T}}^T \underline{\underline{A}} \underline{\underline{T}}$$

$$\underline{\underline{V}} = \underline{\underline{T}}^T \underline{\underline{B}} \underline{\underline{T}} .$$

The matrices $\underline{\underline{T}}$, $\underline{\underline{W}}$ and $\underline{\underline{V}}$ are given for each element type in the succeeding sections.

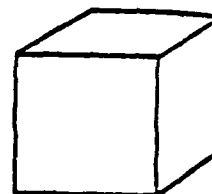
4.21.22 THE EMPTY CUBE (TYPE 0) ELEMENT WITH NO FCSN'S

Standard Cell 0

Empty trilinear cube

Orientation: Arbitrary

Potential Function:



i	N^i
1	$(1-x)(1-y)(1-z)$
2	$(1-z)(1-y)x$
3	$(1-x)y(1-z)$
4	$(1-z)yx$
5	$z(1-y)(1-x)$
6	$x(1-y)(z)$
7	$zy(1-x)$
8	xyz

 w_{ij}

1/3							
0	1/3						
0	-1/12	1/3					
-1/12	0	0	1/3				
0	-1/12	-1/12	-1/12	1/3			
-1/12	0	-1/12	-1/12	0	1/3		
-1/12	-1/12	0	-1/12	0	-1/12	1/3	
-1/12	-1/12	-1/12	0	-1/12	0	0	1/3

 v_{ij}

1/27							
1/54	1/27						
1/54	1/108	1/27					
1/108	1/54	1/54	1/27				
1/54	1/108	1/108	1/216	1/27			
1/108	1/54	1/216	1/108	1/54	1/27		
1/108	1/216	1/54	1/108	1/54	1/108	1/27	
1/216	1/108	1/108	1/54	1/108	1/54	1/54	1/27

4.21.23 THE EMPTY CUBE (TYPE 0) ELEMENT WITH SIX FCSN'S

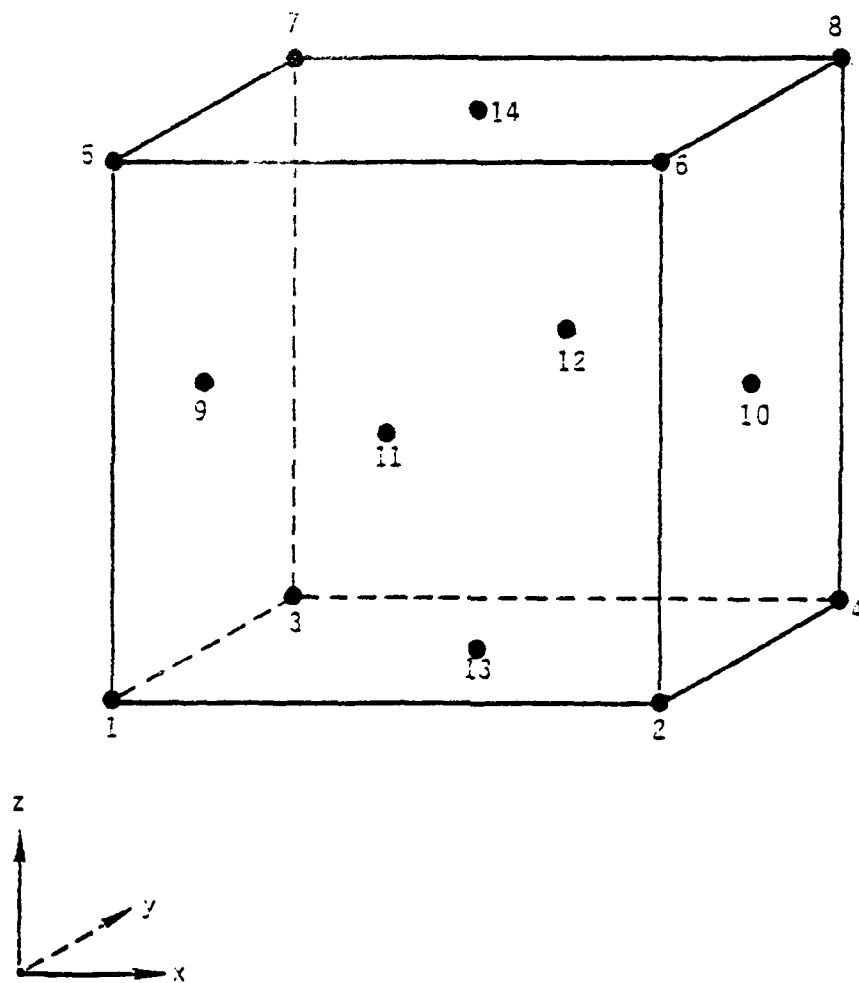


Figure 4.21.23/1. Cubical finite elements with six face-centered surface nodes (FCSNs). The FCSNs are located on the \bar{x} , \bar{y} , \bar{z} faces respectively.

$$\phi(\underline{r}) = \sum_i N^i \psi_i$$

$$0 < x < 1; \bar{x} = 1 - x$$

$$0 < y < 1; \bar{y} = 1 - y$$

$$0 < z < 1; \bar{z} = 1 - z$$

<u>i</u>	<u>Nⁱ</u>	<u>i</u>	<u>Nⁱ</u>
1	$\bar{x} \bar{y} \bar{z}$	9	$16 \bar{x} y \bar{y} z \bar{z}$
2	$x \bar{y} \bar{z}$	10	$16 x y \bar{y} z \bar{z}$
3	$\bar{x} y \bar{z}$	11	$16 x \bar{x} \bar{y} z \bar{z}$
4	$x y \bar{z}$	12	$16 x \bar{x} y z \bar{z}$
5	$\bar{x} \bar{y} z$	13	$16 x \bar{x} y \bar{y} \bar{z}$
6	$x \bar{y} z$	14	$16 x \bar{x} y \bar{y} z$
7	$\bar{x} y z$		
8	$x y z$		

Figure 4.21.23/2. Interpolation functions for the cubical element of Figure 4.21.23/1.

The Matrix T

[illegible]

The Matrix W

+ .79778	+ .48074	+ .41370	+ .48074	+ .41370	+ .43000	-.73037	-.57333	-.73037	-.57333	-.73037	-.57333	-.73037	-.57333
+ .48074	+ .79778	+ .41370	+ .48074	+ .41370	+ .43000	+ .48074	-.73037	-.57333	-.73037	-.57333	-.73037	-.57333	-.73037
+ .48074	+ .41370	+ .79778	+ .48074	+ .41370	+ .43000	+ .43074	+ .41370	-.73037	-.57333	-.73037	-.57333	-.73037	-.57333
+ .41370	+ .48074	+ .48074	+ .79778	+ .43000	+ .41370	+ .48074	-.57333	-.73037	-.57333	-.73037	-.57333	-.73037	-.57333
+ .48074	+ .41370	+ .43000	+ .79778	+ .48074	+ .41370	+ .48074	-.73037	-.57333	-.73037	-.57333	-.73037	-.57333	-.73037
+ .41370	+ .48074	+ .43000	+ .41370	+ .79778	+ .48074	+ .41370	-.57333	-.73037	-.73037	-.57333	-.73037	-.57333	-.73037
+ .41370	+ .43000	+ .48074	+ .41370	+ .48074	+ .79778	+ .41370	-.73037	-.57333	-.73037	-.57333	-.73037	-.57333	-.73037
+ .43000	+ .41370	+ .48074	+ .41370	+ .48074	+ .41370	+ .48074	-.73037	-.57333	-.73037	-.57333	-.73037	-.57333	-.73037
-.73037	-.57333	-.73037	-.57333	-.73037	-.57333	-.73037	-.57333	2.18074	+ .66370	+ .59259	+ .59259	+ .59259	+ .59259
-.57333	-.73037	-.57333	-.73037	-.57333	-.73037	-.57333	-.73037	+ .66370	2.18074	+ .59259	+ .59259	+ .59259	+ .59259
-.73037	-.73037	-.57333	-.73037	-.73037	-.57333	-.73037	-.57333	+ .59259	+ .59259	2.18074	+ .66370	+ .59259	+ .59259
-.57333	-.73037	-.73037	-.57333	-.57333	-.73037	-.73037	-.73037	+ .59259	+ .59259	+ .66370	2.18074	+ .59259	+ .59259
-.73037	-.73037	-.73037	-.73037	-.73037	-.57333	-.73037	-.73037	+ .59259	+ .59259	+ .59259	+ .66370	2.18074	+ .66370
-.57333	-.73037	-.73037	-.73037	-.73037	-.73037	-.73037	-.73037	+ .59259	+ .59259	+ .59259	+ .59259	+ .66370	2.18074

4.2-12

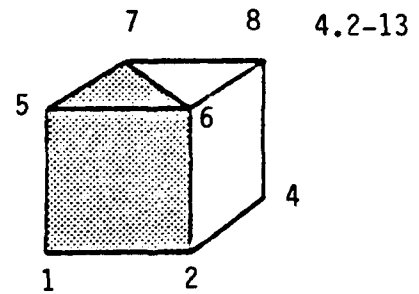
4.2-12

4.21.24 THE WEDGE ELEMENT (TYPE 1)

Half Empty Wedge

$$1 < x+y < 2$$

$$0 < z < 1$$



Node Location

1	0	0	0
2	1	0	0
3	0	1	0
4	1	1	0
5	0	0	1
6	1	0	1
7	0	1	1
8	1	1	1
9	2/3	2/3	0
10	2/3	2/3	1
11	1	1/2	1/2
12	1/2	1	1/2
13	1/2	1/2	1/2

Interpolation Function, N_i

0
$\bar{y} \bar{z}$
$\bar{x} \bar{z}$
$(x-\bar{y}) \bar{z}$
0
$\bar{y} z$
$\bar{x} z$
$(x-\bar{y}) z$
$27\bar{x} \bar{y} \bar{z}(x-\bar{y})$
$27\bar{x} \bar{y} z(x-\bar{y})$
$16(x-\bar{y})\bar{y} z z$
$16(x-\bar{y})\bar{x} z z$
$16\bar{x} \bar{y} z \bar{z}$

where

$$\bar{x} = 1-x$$

$$\bar{y} = 1-y$$

$$\bar{z} = 1-z$$

for Type 1 (Wedge) Element

[illegible]

[illegible]

The Matrix V

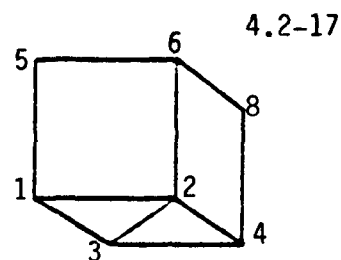
for Type 1 (Wedge) Element

[illegible]

4.21.25 THE TYPE 2 ELEMENT

Cube with Diagonal Line on One Face

Orientation: Line from 2 to 3



Node	Location			Interpolation Function, N_i
1	0	0	0	$(\bar{x}-y)\bar{z}\theta(\bar{x}-y)$
2	1	0	0	$[x\theta(\bar{x}-y) + \bar{y}\theta(x-\bar{y})]\bar{z}$
3	0	1	0	$[y\theta(\bar{x}-y) + \bar{x}\theta(x-\bar{y})]\bar{z}$
4	1	1	0	$(x-\bar{y})\bar{z}\theta(x-\bar{y})$
5	0	0	1	$\bar{x} \bar{y} z$
6	1	0	1	$x \bar{y} z$
7	0	1	1	$\bar{x} y z$
8	1	1	1	$x y z$
9	0	1/2	1/2	$16 \bar{x} \bar{y} y z \bar{z}$
10	1	1/2	1/2	$16 x y \bar{y} z \bar{z}$
11	1/2	0	1/2	$16 x \bar{x} \bar{y} z \bar{z}$
12	1/2	1	1/2	$16 x \bar{x} y z \bar{z}$
13	1/2	1/2	1	$16 x \bar{x} y \bar{y} z$
14	1/3	1/3	0	$27 x y \bar{z}(\bar{x}-y)\theta(\bar{x}-y)$
15	2/3	2/3	0	$27 \bar{x} \bar{y} \bar{z}(x-\bar{y})\theta(x-\bar{y})$

where

$$\bar{x} = 1-x$$

$$\bar{y} = 1-y$$

$$\bar{z} = 1-z$$

The Matrix T

for Type 2 (Special Empty Cube) Element

[illegible]

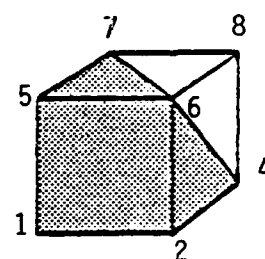
The Matrix W
for Type 2 (Special Empty Cube) Element

1.1617	0.5415	0.5415	0.4155	0.5700	0.4740	0.4749	0.4900	-0.0270	-0.0021	-0.0270	-0.0021	-0.0438	-1.0000	-0.1000
0.5415	1.3519	0.9714	0.5415	0.3599	0.4104	0.3713	0.3599	-0.5177	-0.0075	-0.0075	-0.5177	-0.5077	-1.0109	-1.0109
0.5415	0.9714	1.3519	0.5415	0.3599	0.3713	0.4104	0.3599	-0.0075	-0.5177	-0.5177	-0.0075	-0.5077	-1.0109	-1.0109
0.4455	0.5415	0.5415	1.1017	0.4000	0.4749	0.4749	0.5700	-0.0021	-0.0270	-0.0021	-0.0270	-0.0438	-0.1000	-1.0000
0.5700	0.3599	0.3599	0.4000	0.7954	0.4700	0.4700	0.4105	-0.7313	-0.5724	-0.7313	-0.5724	-0.7209	-0.3009	-0.2059
0.4749	0.4104	0.3713	0.4749	0.4700	0.7949	0.4144	0.4700	-0.5770	-0.7287	-0.7287	-0.5770	-0.7243	-0.2074	-0.2074
0.4749	0.3713	0.4104	0.4749	0.4700	0.4144	0.7949	0.4700	-0.7287	-0.5770	-0.7287	-0.7287	-0.7243	-0.2074	-0.2074
0.4600	0.3599	0.3599	0.5700	0.4105	0.4700	0.4700	0.7954	-0.5724	-0.7313	-0.5724	-0.7313	-0.7209	-0.2059	-0.3009
-0.0270	-0.5177	-0.0075	-0.0021	-0.7313	-0.5770	-0.7287	-0.5724	2.1752	0.0692	0.0017	0.5035	0.5928	0.4000	0.2000
-0.0021	-0.0075	-0.5177	-0.0270	-0.5724	-0.7287	-0.5770	-0.7313	0.0692	2.1752	0.5035	0.0017	0.5928	0.2000	0.4000
-0.0270	-0.0075	-0.5177	-0.0021	-0.7313	-0.7287	-0.5770	-0.5724	0.0017	0.5035	2.1752	0.0692	0.5928	0.4000	0.2000
-0.0021	-0.5177	-0.0075	-0.0270	-0.5724	-0.5770	-0.7287	-0.7313	0.5035	0.0017	0.0692	2.1752	0.5928	0.2000	0.4000
-0.0438	-0.5077	-0.5077	-0.0438	-0.7209	-0.7243	-0.7243	-0.7209	0.5928	0.5928	0.5928	0.5928	2.1490	0.3365	0.3365
-1.0000	-1.0109	-1.0109	-0.1000	-0.3009	-0.2074	-0.2074	-0.2059	0.4000	0.2000	0.4000	0.2000	0.3365	2.0316	0.0000
-0.1000	-1.0109	-1.0109	-1.0000	-0.2059	-0.2074	-0.2074	-0.3009	0.2000	0.4000	0.2000	0.4000	0.3365	0.0000	2.0316

The Matrix V
for Type 2 (Special Empty Cube) Element

0.0223	0.0047	0.0047	0.0110	0.0111	0.0093	0.0093	0.0103	-0.0256	-0.0251	-0.0256	-0.0251	-0.0275	-0.0090	-0.0089
0.0047	0.0217	0.0069	0.0047	0.0025	0.0064	0.0054	0.0025	-0.0162	-0.0088	-0.0088	-0.0162	-0.0167	-0.0038	-0.0038
0.0047	0.0069	0.0217	0.0047	0.0025	0.0054	0.0064	0.0025	-0.0088	-0.0162	-0.0162	-0.0088	-0.0167	-0.0038	-0.0038
0.0110	0.0047	0.0047	0.0223	0.0103	0.0093	0.0093	0.0111	-0.0251	-0.0256	-0.0251	-0.0256	-0.0275	-0.0089	-0.0090
0.0111	0.0025	0.0025	0.0103	0.0217	0.0097	0.0097	0.0064	-0.0168	-0.0230	-0.0168	-0.0230	-0.0172	-0.0086	-0.0106
0.0093	0.0064	0.0054	0.0093	0.0097	0.0220	0.0072	0.0097	-0.0235	-0.0161	-0.0161	-0.0235	-0.0163	-0.0106	-0.0106
0.0093	0.0054	0.0064	0.0093	0.0097	0.0072	0.0220	0.0097	-0.0161	-0.0235	-0.0235	-0.0161	-0.0163	-0.0106	-0.0106
0.0103	0.0025	0.0025	0.0111	0.0064	0.0097	0.0097	0.0217	-0.0230	-0.0168	-0.0230	-0.0168	-0.0172	-0.0106	-0.0086
-0.0256	-0.0162	-0.0088	-0.0251	-0.0168	-0.0235	-0.0161	-0.0230	0.0943	0.0480	0.0602	0.0584	0.0599	0.0367	0.0178
-0.0251	-0.0088	-0.0162	-0.0256	-0.0230	-0.0161	-0.0235	-0.0166	0.0480	0.0943	0.0584	0.0602	0.0599	0.0178	0.0367
-0.0256	-0.0088	-0.0162	-0.0251	-0.0166	-0.0161	-0.0235	-0.0230	0.0602	0.0584	0.0943	0.0480	0.0599	0.0367	0.0178
-0.0251	-0.0162	-0.0088	-0.0256	-0.0230	-0.0235	-0.0161	-0.0166	0.0584	0.0602	0.0480	0.0943	0.0599	0.0178	0.0367
-0.0275	-0.0167	-0.0167	-0.0275	-0.0172	-0.0163	-0.0163	-0.0172	0.0599	0.0599	0.0599	0.0599	0.0992	0.0213	0.0213
-0.0090	-0.0038	-0.0038	-0.0089	-0.0088	-0.0106	-0.0106	-0.0106	0.0367	0.0178	0.0367	0.0178	0.0213	0.0439	0.0000
-0.0089	-0.0038	-0.0038	-0.0090	-0.0106	-0.0106	-0.0106	-0.0088	0.0178	0.0367	0.0178	0.0367	0.0213	0.0000	0.0439

4.21.26 THE TETRAHEDRON ELEMENT



Tetrahedron (Type 3)

$$2 < x+y+z < 3$$

Node	Location			Volume	N_i
1	0	0	0	0	0
2	1	0	0	0	0
3	0	1	0	0	0
4	1	1	0	.007756	$1 - z$
5	0	0	1	0	0
6	1	0	1	.007756	$1 - y$
7	0	1	1	.007756	$1 - x$
8	1	1	1	.009542	$x + y + z - 2$
9	1	2/3	2/3	.032101	$27 \bar{x} \bar{y} \bar{z} (x + y + z - 2)$
10	2/3	1	2/3	.032101	$27 \bar{x} \bar{y} \bar{z} (x + y + z - 2)$
11	2/3	2/3	1	.032101	$27 \bar{x} \bar{y} \bar{z} (z - \bar{x} - \bar{y})$
12	2/3	2/3	2/3	.037552	$27 \bar{x} \bar{y} \bar{z}$

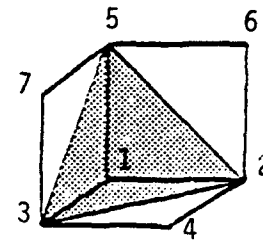
For the Type 3 (Tetrahedron) Volume Element

4.2-22

For the Type 3 (Tetrahedron) Volume Element

4.2-23

4.21.27 THE TRUNCATED CUBE ELEMENT (TYPE 4)



Truncated Cube (Type 4)

$$1 < x+y+z < 3$$

Node	Location			Volume	N_i
1	0	0	0	0	0
2	1	0	0	-.0380	$(2x - y - z + 1) K(2 - x - y - z)/3$
3	0	1	0	-.0380	$(2y - x - z + 1) K(2 - x - y - z)/3$
4	1	1	0	.0406	$(1 + x + y - 2z) \theta_2/3 + (1-z)\theta_3$
5	0	0	1	-.0380	$(2z - x - y + 1) K(2 - x - y - z)/3$
6	1	0	1	.0406	$(1 + x + z - 2y) \theta_2/3 + (1-y)\theta_3$
7	0	1	1	.0406	$(1 + y + z - 2x) \theta_2/3 + (1-x)\theta_3$
8	1	1	1	-.0709	$(x + y + z - 2) \theta_3$
9	0	2/3	2/3	.1119	$27 \bar{x} \bar{y} \bar{z} K(y + z - 1)$
10	1	1/2	1/2	.1499	$16 x y \bar{y} z \bar{z}(x + y + z - 1)$
11	2/3	0	2/3	.1119	$27 \bar{x} \bar{y} \bar{z} K(x + z - 1)$
12	1/2	1	1/2	.1499	$16 x \bar{x} y z \bar{z}(x + y + z - 1)$
13	2/3	2/3	0	.1119	$27 \bar{x} \bar{y} \bar{z} K(x + y - 1)$
14	1/2	1/2	1	.1499	$16 x \bar{x} y \bar{y} z(x + y + z - 1)$
15	1/3	1/3	1/3	.1112	$K(1 - x - y + 2z) K(1 - y - z + 2x)$ $K(1 - x - z + 2y) K(2 - x - y - z)$

$$K(s) = s \theta(s)$$

$$\bar{x} = 1 - x$$

$$\theta_2 = (x + y + z - 1) \theta(2 - x - y - z) \quad \bar{y} = 1 - y$$

$$\theta_3 = (x + y + z - 2) \theta(x + y + z - 2) \quad \bar{z} = 1 - z$$

For the Type 4 (Truncated Cube) Volume Element

[illegible]

4.21.28 THE SLANTED THIN PLATE ELEMENT (TYPE 5)

The type 5 element is treated as two type 1 (4.21.24) elements.

4.22 BOUNDARY CONDITIONS

The potential solver can use either fixed potential or fixed normal electric field. Usually, fixed potentials are used for all surfaces, but when hopping secondary currents are present fixed normal electric field boundaries on insulating surfaces may be necessary. Conducting surfaces always have fixed potential boundary conditions.

Fixed normal electric field boundary conditions are necessary in the presence of large photoemission or hopping secondary electron currents. The relation between the normal and parallel electric fields in the presence of large secondary electron conductivities is

$$E_{\perp} \Delta x = \sqrt{4\langle\epsilon\rangle} E_{\parallel} \Delta x$$

where Δx is the mesh spacing and $\langle\epsilon\rangle$ is the mean secondary electron energy (eV).

The condition of local current balance for these surfaces is

$$0 = J_{\text{wfd}} - \nabla \cdot J_{\text{hc}}$$

where J_{wfd} represents the weakly-fielded-dependent current (of electrons incident on the surface cell) and J_{hc} is the hopping current across the edge next to the most relatively positive neighboring cell. Writing

$$J_{\text{hc}} = \sigma_{\parallel} E_{\parallel}$$

and

$$\sigma_{\parallel} = - \frac{4\langle\epsilon\rangle J}{E_{\perp}^2} e$$

with $J_e < 0$ the low-energy electron emission current, the current balance equation becomes

$$E_{\perp} = \sqrt{4\langle\epsilon\rangle Y(-\Delta\cdot E_{\parallel})}$$

where Y is the secondary yield.

If the fixed field condition is used, the surface potential will be calculated by the potential solver instead of by the surface charging module, CHARGE. The charging module makes the initial decision of which boundary condition is appropriate for each surface. If the potential solver later finds the secondary yield for the insulator with a fixed field boundary condition, to be less than one, it will change to constant potential boundary condition and set the surface potential to V_s where

$$V_s = \frac{kT}{2} \log \frac{m_{\text{electron}}}{m_{\text{ion}}}$$

4.30 MATRIX SOLVERS

POLAR (NTERAK) currently uses two methods to solve matrix equations of the form

$$\underset{\sim}{M} \underset{\sim}{X} = \underset{\sim}{d}$$

where $\underset{\sim}{M}$ is a given matrix, $\underset{\sim}{d}$ is data and $\underset{\sim}{X}$ is the solution vector. These methods are the Conjugate Gradient Method and the Incomplete Cholesky Conjugate Gradient Method (ICCG).

4.31 CONJUGATE GRADIENT METHOD

This method is used for the Poisson equation solution (4.20, 4.44) where $X = \phi$ (potential) which can have 10,000 or more components. The data requirements are made tangible by performing the solution element by element. This is expressed by Eq. (4.6) of Section 4.21,

$$\sum_e \sum_j \left[(W_{ij}^e - \frac{\rho'}{\epsilon} V_{ij}^e) \phi_j - \frac{\rho_o}{\epsilon} R_i^e \right] = 0 \quad \text{for each } i \in e \quad (4.6)$$

To simplify notation we will perform the sum over elements and treat the problem as a whole, thus,

$$\sum_e (W_{ij}^e - \frac{\rho'}{\epsilon} V_{ij}^e) = \underset{\sim}{M} \quad ,$$

$$\sum_e \frac{\rho_o}{\epsilon} R_i^e = \underset{\sim}{Q} \quad ,$$

and

$$\phi_j = \phi$$

Thus we have

$$\underset{\sim}{M} \phi - \underset{\sim}{Q} = 0 \quad (4.7)$$

We may solve Eq. (4.7) iteratively. Our initial choice of ϕ will yield a residual r

$$\underline{\underline{M}} \phi - Q = -r$$

The iterative scheme used is the Conjugate Gradient technique. It is based on the following equations:

$$r_0 = Q - \underline{\underline{M}} \phi^0$$

$$u^0 = r^0$$

$$a^i = (r^i, r^i) / (u^i, \underline{\underline{M}} u^i)$$

$$\phi^{i+1} = \phi^i + a^i u^i$$

$$r^{i+1} = r^i - a^i \underline{\underline{M}} u^i$$

$$b^i = (r^{i+1}, r^{i+1}) / (r^i, r^i)$$

$$u^{i+1} = r^{i+1} + b^i u^i$$

These equations may be iterated upon until the resultant ϕ vector becomes the solution to Poisson's equation.

The major computational operation in the iterative set of equations is the evaluation of the matrix-vector product $\underline{\underline{M}} u$. The vectors ϕ , u , and r all have the same number of grid points. M contains the square of this number. Such a huge array is impractical to store all at once and so $M u$ is evaluated using the following implicit algorithm

$$r = \sum_e r_e - \sum_e w_e u$$

The $w_{\tilde{e}}$ matrices are of reasonable dimension, for example, 8×8 for an empty cube. The residual r is constructed element by element and then summed. These "weight" matrices $w_{\tilde{e}}$ may be calculated analytically for each type of empty or partially filled volume element, allowed by POLAR. There are seven of these. Filled cells are not included in the potential calculation. This is how POLAR treats filled, partially filled and empty elements, differently.

4.32 ICCG, THE INCOMPLETE CHOLESKY CONJUGATE GRADIENT METHOD

ICCG is used to solve the charging equations (4.50, 5.70), where $X = V_S$, surface voltages. Components of V_S generally number 1000 or less. This allows the entire problem to be kept in memory at one time by saving only the non-zero elements of M (5.73.2). ICCG will find series of approximate inverses for M finding X as

$$\tilde{x} = \tilde{M}^{-1} d .$$

It is iterative, but iterates on \tilde{M}^{-1} as well as \tilde{X} . A more complete description will be found here in future revisions. Reference 4-5: Kershaw, D. (1978), "The Incomplete Cholesky-Conjugate Gradient Method for the Iterative Solution of Systems of Linear Equations," Journal of Computational Physics, 26, p. 43.

4.40 SPACE CHARGE AND CURRENT COMPUTATION

This section describes the numerical techniques such as integration, that are used to effect the models described in Chapter 3. In some cases, the computational requirements are satisfied by the top-down structuring of simple subroutines. In these cases, the Chapter 4 discussion is deferred to Chapter 5 to avoid repetition.

4.41 WEAK FIELD IONS, PRESHEATH AND WAKE

Initial ion densities are calculated before particle tracking using geometric shadowing corrected by the electric field effects of the electrons and ions. As discussed in Section 3.31, the neutral ion approximation can be used as a starting estimate of the ion density. The coding which calculates the neutral ion densities is described in Section 5.61.10. The coding and equations used to model the electric field correction for neutral ions is presented in Section 5.61.20.

4.42 THE POLAR SHEATH MODEL (TECHNICAL)

Section 3.60 contains a general discussion of the POLAR sheath model. This section is not designed to be a complete discussion, but to fill in the technical details of the model.

4.42.1 SHEATH EDGE ALGORITHM (SHEATH)

The SHEATH routine takes as input a sheath edge potential and the eight vertex potentials of a single cubic element, $P(2,2,2)$. It determines whether the sheath potential contour passes through the specified element and if it does it generates a number of triangles, NPART, with areas $W(I)$ and center $X(3,I)$ which approximates the equipotential surface.

The sheath location proceeds by finding edges whose vertex potentials bracket the sheath potential. If none do, NPART is set to zero and control returns to the calling program. For the cases of three intersections a single triangle is constructed from the intersection points with the area calculated by the Triangle AREA routine. For four or five edge intersections the centroid of the intersection points is found and triangles constructed using adjacent intersection points and the centroid. Thus for four edge intersections, four triangles are formed. SHEATH then would return $NPART = 4$ and the center coordinates of each of the four triangles.

4.42.2 CURRENTS TO THE SHEATH SURFACE

Ions

The current density to the sheath edge is calculated by assuming the sheath to be a perfectly absorbing spherical surface in a flowing plasma. The potential around the sphere is assumed to be spherically symmetric and

attracts ions. The coordinate systems utilized are indicated in Figure 4.42/1 below (shown twice to reduce cluttering)

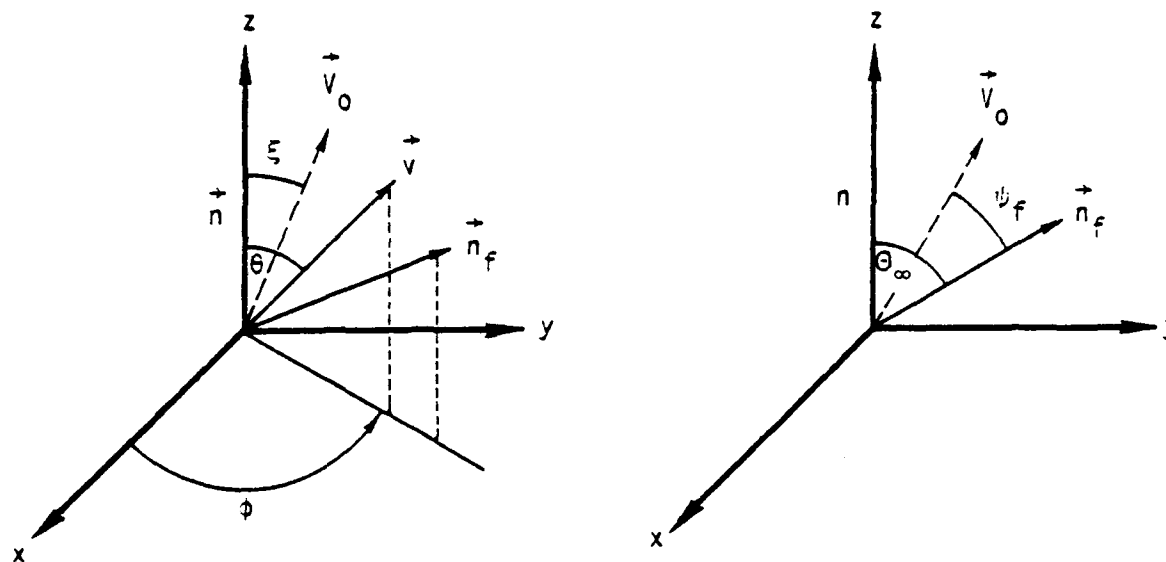


Figure 4.42/1.

The following definitions are used:

a = sphere radius

\vec{V}_0 = satellite velocity

$\Phi(r)$ = potential energy of ion at r

\vec{n} = unit vector at position \vec{r} on sphere where normal current density is to be calculated

\vec{n}_f = unit vector in final direction (at $r = \infty$) mapped by particles launched from $\vec{r} = (a, \vec{n})$ with velocity \vec{v}

x-z plane = plane determined by \vec{n} and \vec{V}_0

$$\begin{aligned}
 \xi &= \text{angle between } \vec{n} \text{ and } \vec{V}_o \\
 \theta_\infty &= \text{polar angle of particle for } (n = \infty, n_f) \\
 \phi &= \text{angle between x-z plane and orbital plane} \\
 f_o(\vec{V}_o) &= (2\pi v_T^2)^{-3/2} \left\{ \exp - (\vec{V}_o - \vec{V}_o)^2 / 2v_T^2 \right\} \\
 m &= \text{ion mass} \\
 v_T^2 &= kT/m
 \end{aligned}$$

For a particle moving in a central potential the conserved quantities are ($e = m = 1$):

$$\begin{aligned}
 \frac{1}{2} v^2 + \Phi(a) &= \frac{1}{2} v_o^2 = E && \text{Energy} \\
 L &= v \cdot a \cdot \sin\theta && \text{Angular Momentum} \\
 \phi &&& \text{Azimuth}
 \end{aligned}$$

The normal current density $j(\vec{r})$ at a point $\vec{r} = (a, \vec{n})$ on the sphere is given by

$$\begin{aligned}
 j &= j(a, \vec{n}) = \int (\vec{v} \cdot \vec{n}) f_o(\vec{V}_o) d^3\vec{v} \\
 &= (2\pi v_T^2)^{-3/2} \int_0^\infty dV \int_0^{2\pi} d\theta \int_0^{\theta_{\max}} d\theta \left[\exp \frac{1}{2} (V_o^2 - 2V_o v_o \cos \phi + V_o^2) / v_T^2 \right] \\
 &\quad \times v^3 \cos\theta \sin\theta
 \end{aligned}$$

where

$$\cos\phi_f = \cos\xi \cos\theta_\infty - \sin\xi \sin\theta_\infty \cos\phi$$

$$\theta_\infty(v_o, \theta) = a \sin\theta \int_a^\infty \frac{dr}{r^2} \left[1 - \frac{a^2}{r^2} \sin^2\theta + \frac{\Phi(a) - \Phi(r)}{\frac{1}{2} v_o^2 - \Phi(a)} \right]^{-1/2}$$

Performing the ϕ integration, which can be done analytically, and using energy conservation,

$$j = (2\pi v_T^2)^{-3/2} \int_0^\infty v_o (v_o^2 - 2\Phi(u)) e^{-v_o^2/2v_T^2} F(v_o) dv_o$$

$$F(v_o) = 2\pi \int_0^{\theta_{\max}(v_o)} e^{-v_o v_o \cos\xi \cos\theta_s / v_T^2} e^{-v_o^2/2v_T^2}$$

$$\cdot I_0 \left(\frac{v_o v_o \sin\xi \sin\theta_s}{v_T^2} \right) \sin\rho \sin\theta d\theta$$

where I_0 is the modified Bessel function of zero order.

For numerical calculations, define $t = x/3.75$ and approximate I_0 by

$$I(x) = 1 + 3.5156229 t^2 + 3.0899424 t^4 + 1.2067492 t^6$$

$$+ 0.2659732 t^8 + 0.0360768 t^{10} + 0.0045813 t^{12} + \epsilon$$

$$\epsilon < 1.6 \times 10^{-7}$$

for $-3.75 \leq x \leq 3.75$

$$x^{1/2} e^{-x} I_0(x) = .39894228 + .01328592 t^{-1}$$

$$+ .00225319 t^{-2} - .00157565 t^{-3}$$

$$+ .00916281 t^{-4} - .02057705 t^{-5}$$

$$+ .02635537 t^{-6} - .01647633 t^{-7}$$

$$+ .00392377 t^{-8} + \epsilon$$

$$\epsilon < 1.9 \times 10^{-7}$$

for $3.75 \leq x < \infty$

For an inverse square potential, $\theta_{\max} = \pi/2$, and the change in polar angle can be found analytically

$$\theta_s = \frac{\sin\theta}{(\sin^2\theta - b^2)^{1/2}} \sin^{-1} \left(\frac{\sin^2\theta - b^2}{1 - b^2} \right)^{1/2} ; \sin^2\theta - b^2 \geq 0$$

$$= \frac{\sin\theta}{(b^2 - \sin^2\theta)^{1/2}} \sinh^{-1} \left(\frac{b^2 - \sin^2\theta}{1 - b^2} \right)^{1/2} ; \sin^2\theta - b^2 < 0$$

where

$$b^2 = \frac{|\Phi(a)|}{E + |\Phi(a)|} < 1 \quad (0 < E < \infty)$$

$$E + |\Phi(a)| = \quad -$$

$$E = \frac{1}{2} v_o^2$$

$$\sinh^{-1}x = \ln \left[x + \sqrt{x^2 + 1} \right]$$

Note for $1 - b^2 \ll 1$, θ_s may be $> 2\pi$, i.e., the particles may execute a spiralling orbit. Accuracy may require that δv_o , $\delta\theta$ not produce large $\delta\theta_s$.

The above computation is used to calculate the flux through a portion of the sheath edge found using the algorithm discussed in Section 4.42.1. This allows the current through the sheath to be broken into small, flat triangles. These triangles are the "particles" which are pushed by the particle pushing routines in CURREN.

The particles are defined to be located at the triangle's centroid and to have an initial velocity into the sheath. The initial velocity of the ions accounts for contributions from the Mach and thermal velocity. Particle pushing is done in the spacecraft reference frame so the plasma appears to be flowing with the Mach velocity. Since portions of the sheath will have surface normals in the downstream direction, shadowing by the sheath needs to be taken into account for both the current and initial velocity associated with a given particle. The

particle currents were discussed above. The initial velocity needs to be found so that the mean particle velocity through a surface is well represented and varies continuously from the upstream to downstream portions of the sheath. The velocity should also approach the thermal velocity at low Mach velocities. POLAR uses the following to initialize particle velocities:

For spacecraft velocities less than $0.1 * v_{th}$, where v_{th} is the ion thermal velocity, the initial velocity, v_o , is

$$\vec{v}_o = v_{th} \hat{E}$$

\hat{E} is the direction of the inward electric field at the sheath. When the Mach vector is greater than $0.1 * v_{th}$, the following is used:

$$v_o = \begin{cases} (\hat{E} \times \vec{V}_m) \times \hat{E} + v_{th} \hat{E} & \hat{E} \cdot \vec{V}_m < 0 \text{ (downstream)} \\ \vec{V}_m + \frac{|\hat{E} \times \vec{V}_m|}{|\vec{V}_m|} v_{th} \hat{E} & \hat{E} \cdot \vec{V}_m \geq 0 \text{ (upstream)} \end{cases}$$

with \vec{V}_m being the Mach velocity of the flowing plasma.

An unfortunate side effect of starting particles with a velocity strongly dependent on the surface normal is that the angular distribution of thermal velocities is not modeled. Particularly symmetric problems (e.g., long, cylindrical wakes at high Mach velocities) tend to experience excessive focusing where particle trajectories are trapped. To combat this problem, particles can be broken into a number of smaller particles. Each particle is started with a different initial velocity to generate an angular distribution of thermal velocities. See Section 3.43 for more information on how the particles are spread.

Electrons

When the attracted species are electrons, the initial currents and velocities are calculated differently. The sheath electron flux is the one-sided thermal flux from the presheath, quasi-neutral plasma. The orbit limited ion density at the sheath is found using the same techniques as described in Section 3.31. (See also Section 4.52.7.) The initial velocity is defined to be the average thermal velocity through the sheath in the direction of the inward electric field.

In the presence of magnetic fields, the electron sheath fluxes are restricted by the magnetic flux tube. This is modeled by the following factor

$$j_e' = j_e (0.1 + 0.9 | \hat{n} \cdot \hat{B} |)$$

where j_e' is the restricted flux, j_e is the nonmagnetic limited flux, \hat{n} is the electron sheath surface normal, and \hat{B} is the direction of the magnetic field. (See also Section 6.43.20.)

4.42.3 SHEATH PARTICLE ASSIGNMENT

Each volume element of a problem is inspected for the presence of the sheath edge equipotential as described in Section 4.42.1. Each triangular subsurface of the sheath is a potential sheath particle; however, many times a portion of an equipotential is not really a source of current. This is assumed to occur when there exists just "outside" the sheath (in the direction opposite of \vec{E}) a portion of the object, or a high potential region of the opposite sign. Both of these conditions are checked by calculating the inward initial velocity for the particle as per Section 4.42.2, reversing it, and tracking the particle backwards through two volume elements. If no "obstacles" are found, the particle is assigned a current or weight and placed in a particle list. This list is later read and the trajectories advanced as described in Sections 4.42.4 and 5.62. Finally, the sheath currents are not calculated for each particle but interpolated from a pre-calculated table of values (Section 4.42.2).

4.42.4 TRAJECTORY TRACKING

Trajectory tracking can be an expensive endeavor, and a source of unpredictable error. To combat these problems, POLAR uses two different methods to follow ion and electron trajectories.

When pushing ions, the full step method is used in empty elements (those which do not touch the object) where complex \vec{E} fields are not anticipated. For these elements, \vec{E} at the cell center is used for the entire cell, and a single step parabolic trajectory is calculated for the element. This is accomplished by analyzing independently the three components of the equation

$$\vec{x}_i = \vec{x}_{i-1} + \vec{v}_{i-1}t + \frac{1}{2} \frac{q}{m} \vec{E} t^2 \quad (1)$$

for the shortest time, t , that a particle needs to reach an element face. Negative, imaginary and zero times are rejected. A number of special conditions may occur involving round-off errors and the traversal of exceedingly small paths in the corner of elements. The treatment of these problems are discussed in detail in Section 5.62.22. Following the choice of the shortest valid time, the trajectory is advanced to another (or possibly the same) element face and the new velocity vector is calculated at \vec{x} and is

$$\vec{v}_i = \vec{v}_{i-1} + \frac{d\vec{v}}{dt} t$$

where

$$\frac{d\vec{v}}{dt} = \frac{q}{m} \vec{E} + \vec{v} \times \frac{\vec{B}_G}{c}$$

with \vec{B}_G being the magnetic field vector in gauss. The magnetic field effects on the particle velocity are implemented as a rotation of the velocity vector after the acceleration due to the average electric field is added. (This is discussed in more detail in 5.62.22.)

The total energy is checked at the new position against the original value. The new total energy is $1/2 m\vec{V}^2 + eV(\vec{x})$, where $V(\vec{x})$ is the bilinear potential calculated for the exit location on the exit face of the element. The energy is renormalized by adjusting the magnitude of \vec{V} without modifying its direction.

For volume elements that border the object, more complex E fields are anticipated so POLAR uses a slower, but more accurate, step-push method where Eq. (1) is integrated using timesteps estimated to be approximately 0.1 of the element traversal time. At each step, \vec{E} is determined by analytically differentiating the trilinear potential function (Section 4.20). The step-push method and the routines that affect it are discussed in greater detail in Section 5.62.22.

The pushing of electrons requires two different pushing routines. The electrons use the equivalent of MOVER, EMOVE to move to the face of an element and ESTEP to make short steps within an element much the way ions are small stepped by STPPSH. Because the gyroradius of electrons can be comparable to or even smaller than a mesh spacing, electrons use ESTEP when the Larmor radius is large. EMOVE is used when the guiding center approximation of the electron trajectory is appropriate (5.62.22).

POLAR's sliced grid system (Section 4.11) forces additional computational considerations on the trajectory tracking because only a small set of potentials are stored in core at any one time. Potentials are paged in and out in slices at nodes of a constant z value. As a result, trajectory tracking is controlled by a "pusher" that sweeps back and forth in z , advancing all trajectories through the space between z and $z+1$. Trajectories moving opposite the pusher are written out to disk and picked up on the return pass (Section 5.62.22). Although this complicates the coding somewhat, there is a gain in efficiency, and a bonus in that trapped orbits can be simply controlled by limiting the passes of the pusher.

4.42.5 SHEATH ION DENSITIES

Once a sheath edge surface has been located and subdivided (Section 4.42.1), the input current, J_i , calculated (Section 4.42.2) and that current assigned to a representative particle, g , (Section 4.42.3), the particle trajectory is followed inward as described in Section 4.42.4. Ion densities, n_i , are determined in each cell by observing that if each trajectory, j , represents a constant current $J_{ij} = dq_{ij}/dt$, each trajectory makes a contribution to the overall element density of

$$\Delta n_{ij} = \frac{J_{ij} \Delta t}{\text{element volume}}$$

where Δt is the time required to cross the volume element. The total density is just

$$n_i = \sum_j \Delta n_{ij} .$$

This has been dubbed the method of weighted deposition (Ref. 3-5). It can be seen that acceleration effects and convergence effects are accounted for by the Δt , and the \sum_j respectively.

This method demands a large number of particles for good statistics and we have found that the three to six particles per sheath volume element, chosen by the sheath edge algorithm (Section 4.42.1), work quite well. Problems in accuracy can still be anticipated when there exists repulsive regions within a sheath, or when the method is being incorrectly applied to an orbit-limited problem where particles might numerically diffuse into allowed trapped orbits. In this case repeated orbits would give erroneously high densities. Even in strongly space charge-limited sheaths, unusual geometry could lead to trapped orbits, so POLAR sets a user controlled limit on the number of front-to-back pushing sweeps (Section 6.43.20) to control this problem.

Finally, these sheath ion densities are known as RHOI's in POLAR and are calculated by the CURREN segment of NTERAK. The ultimate use of these densities in the Poisson solution are discussed in Section 4.43.2.

4.43 CHARGE DENSITY

POLAR iterates between calculating the potential for fixed charge densities, and calculating charge densities for fixed potentials. The potentials are calculated iteratively, starting from the last potential used to calculate the charge densities. While the calculated charge densities are physically consistent with the first potential iterate, there is no guarantee that they remain that way during the iteration process. Equilibrium plasmas respond to potentials by shielding; that is the plasma spacecharge has the opposite sign as the applied potential. The QSCRN algorithm examines the potentials and densities element by element, and ensures that the charge density used in each iteration is physically reasonable. Unphysical conditions arise when the sheath boundary moves during an iteration. When this occurs, QSCRN substitutes a charge density based on Debye shielding for the out of date particle pushed density.

QSCRN also restricts the magnitude of the charge density used in a particular element to that consistent with the zone size. Since the plasma is shielding, the charge density is restricted to be less than that which will change the sign of the potential in the zone. In practice, this restriction is relaxed, and the charge limited to SQALPH times the maximum charge, where SQALPH is typically chosen to be about 3. Too large a value for SQALPH leads to oscillations at the sheath edge.

4.43.1 ELECTRONS

Two different methods of finding the electron space charge densities are used; one for positive potential regions within the sheath and one for the other cases. When electrons are repelled, the charge density is approximated by an isotropic Boltzmann equilibrium distribution, i.e.,

$$n_e = n_0 \exp(eV(\vec{x})/kT) .$$

This approximation may be invalid near weakly repelling surfaces, space potential barriers, and in magnetically insulated regions. Other conditions may arise where the electron distribution will not be isotropic and a Boltzmann distribution would not be justified. When space potentials are positive enough to define an electron collecting sheath, the electron space charge is calculated by pushing particles. These densities are found much the way pushed ion space charge densities are computed (see Ion Charge Density, 4.43.2). The pushed electron current is called RHOE in the POLAR code.

4.43.2 ION CHARGE DENSITY

Ions are generally considered to be the attracted specie in POLAR. This, plus an allowance for possibly high Mach numbers, means that ion densities may not be determined by any local approximation. Thus ion densities are currently determined by two methods: Weak field ions (Section 4.41) known in the POLAR coding as GI's (geometric ions) and GH's (geometric hydrogen); and sheath ion densities (Section 4.42) known in the coding as RHOI's.

GI's and GH's are determined at the onset of a calculation and remain unchanged thereafter. The RHOI's and RHOE's are calculated whenever a CURREN step (the sheath ion tracking process) is called for. Immediately following a CURREN step, POLAR creates from these two data sets, an ultimate ion density list, the DION's that are used in the Poisson calculation. This is done by choosing RHOI's for elements inside the sheath, and GI's for points outside. For elements containing a portion of the sheath surface, the GI value is used instead of the RHOI. Similarly, the electron density list, DELC, is created using the pushed electron densities, RHOE, inside of the electron sheath and quasineutral electron densities outside.

4.44 THE CHARGE STABILIZED POISSON ITERATION

The Poisson equation can be written dimensionlessly as

$$-\Delta^2 \phi = L^{-2} (n_i - n_e) \quad (1)$$

where

$$\phi = eV/kT, L^2 = \epsilon_0 kT/N_0 e^2 h^2 = \lambda^2/h^2$$

is the dimensionless Debye length, N_0 is the ambient density, $n_i = N_i/N_0$, $n_e = N_e/N_0$, and the Laplacian is also normalized by h^2 . The calculation of n_i and n_e is discussed in Section 4.43. POLAR solves this equation on its discrete mesh of uniform spacing h , using the finite element method described in Section 4.21.

The traditional approach to the solution of equation (1) has been an explicit iteration of the form

$$-\Delta^2 \phi^\nu = L^{-2} [n_i(\phi^{\nu-1}) - n_e(\phi^{\nu-1})] \quad (2)$$

where ν is the iteration index, and the charge density is determined using the potentials of the previous iteration. This method can be shown to be unstable (ref. 4-2) when the Debye length, λ , becomes small with respect to other scale lengths of the problem. This can be understood by considering that a smooth potential variation over a distance of, say, 1000λ , would require a smooth $\Delta^2 \phi$ (the 'second derivative') which is in turn given everywhere by the charge density. But, maintaining a smooth charge density distribution is difficult when any errors in determining $(n_e - n_i)$ are multiplied by a huge L^{-2} . There is one effective remedy to this dilemma due to Parker (ref. 4-2), but the process reported here appears to be more efficient in the short Debye length limit. This method involves the combination of two concepts. One uses a partial implicitization of the repelled density (n_e , here) (ref. 4-3). The other simply reduces the charge density to an acceptable level whenever the first method is inadequate.

Suppose a plasma of ambient density N_0 and temperature T consists of Boltzmann electrons, $N_e(\vec{r}) = N_0 \exp(\phi(\vec{r}))$ and ions of known density $N_i(\vec{r}) = N_0 n_i(\vec{r})$. The normalized charge density is then given by

$$q(\vec{r}, \phi^\nu(r)) = L^{-2} [n_i(\vec{r}) - \exp(\phi^\nu(\vec{r}))] \quad (3)$$

Equation (3) may be linearized about the previous potential iterate

$$q(\phi^\nu) \approx q(\phi^{\nu-1}) + q'(\phi^{\nu-1}) * (\phi^\nu - \phi^{\nu-1})$$

where $q' = \partial q / \partial \phi$, and the \vec{r} dependence has been dropped for clarity. With this expression we may write the implicit Poisson iteration scheme

$$-\nabla^2 \phi^\nu - q'(\phi^{\nu-1}) * \phi^\nu = q(\phi^{\nu-1}) - q'(\phi^{\nu-1}) * \phi^{\nu-1} \quad (4)$$

Though it is not immediately obvious, the implicit character of (4) makes it more stable than scheme (2). This can be understood by realizing that in equation (3) the electron density was treated as an independent variable, whereas in (4) the electron density is determined simultaneously with the potential, both being consistent with the ion density.

The finite element approximation (Section 4.21) to (4) produces the matrix equation

$$\sum_e (\tilde{W}^{(e)} - \tilde{S}^{(e)} \tilde{Y}^{(e)}) * \phi^\nu = S - \tilde{S}^{(e)} * \phi^{\nu-1} \quad (5)$$

where S is derived from q by the following analysis:

For $L \gg 1$, S is simply the total charge associated with each node, q . However, for $L \ll 1$, numerical noise and features like a sheath edge which may span only a few λ , become incorrectly amplified when the q determined at a point becomes multiplied by the entire nodal volume. When it is not possible to reduce the zone size, stability can be preserved by replacing Q (and Q') with a reduced value S (S') which is calculated to be the maximum allowable charge for the element.

Because of the artificial amplification argument, S is often the more realistic total for an element. Before deriving S , we define the barometric potential $\phi_b = \ln(n_i)$ potential for which $Q = 0$ and note that it is important that $S + Q$ as $\phi + \phi_b$ if quasineutral regions are to be modeled correctly. To determine S , consider a capacitor with potential difference $(\phi_b - \phi)$, area h^2 , and a separation of h . The charge q_c on this capacitor is given by

$$q_c = C\Delta V = \frac{\epsilon_o h^2}{he} (\phi_b - \phi) kT$$

In the units of our previous q , q_c becomes

$$q_L = \alpha(\phi_b - \phi) = \alpha(\phi_b - \phi) \quad (6)$$

which is the maximum allowable charge per element, with the parameter α , adjusted to insure that q_L is maximized. Thus at each node, we choose for the charge

$$|S| = \min(|q_L|, |q|)$$

with

$$S' = \begin{cases} -\alpha & \text{for } S = q_L \\ L^{-2} \exp \phi & \text{for } S = q \end{cases}$$

Actually, S and S' are smoothed in POLAR to decrease numerical noise and spatial potential oscillations. This involves forming the linear screening term

$$SCRN = S'(\bar{\phi}) + \bar{S}'/2$$

where

$$\bar{S}' = \sum_I V(I) \cdot S(\phi(I))$$

where I indexes the nodes of an element and $V(I)$ is a nodal volume normalized by a cubic element volume of one; similarly for ϕ . Also smoothed is the nodal charge $Q(I)$,

$$Q(I) = \frac{1}{2} (S(\bar{\phi}) + S(\phi(I)))$$

Additionally, POLAR can output the value $S(\bar{\phi})$ centered charge actually chosen by this algorithm.

Finally, the $Q(I)$, and SCRN are used for S and $\bar{S}^{(e)}$ in Eq. (5).

The effect of this algorithm is this: If a problem has been specified where a boundary potential would be screened in less than a zone or two (the limit of any code's resolution), sufficient sheath charge will be redistributed so as to allow the potential to be screened over the minimum number of zones that are consistent with stability.

The charge stabilization algorithm is effected by the subroutine, QSELT and QSCRN which are further described in Section 5.50.

4.44.1 SHEATH IONIZATION EFFECTS ON SPACE CHARGE

A secondary effect arising from the collection of electrons is the ionization of neutrals within the sheath. The electrons are immediately collected by the positive surfaces. The ions however, owing to their greater mass, migrate more slowly out of the sheath. These ions produce space charge which cancels part of the space charge due to the attracted electrons. The effect is to reduce the screening of the positive surfaces resulting in a larger sheath. The degree to which sheath ionization enlarges the sheath depends on the generation rate of ions within the sheath. This in turn depends on the ionization cross-section of the neutrals, the incident flux of electrons, and the path length for ionization. The enlargement of the sheath due to ionization can be characterized by the ratio of the flux of ions out of the sheath to the flux of electrons incident on the sheath. Increasing this ratio increases the size of the sheath up to a limiting ratio after which the

sheath becomes unstable. At this point the sheath grows with time and the static ideas of a sheath break down.

The sheath instability point can be determined in one-dimension by writing Poisson's equation including both the electrons and generated ions

$$\epsilon_0 \Delta^2 \phi = - (J_{ion}/2e(V-\phi)/m_{ion})^{1/2} + (J_{elec}/(2e\phi/m_{elec}))^{1/2}$$

Integrating both sides from 0 to the sheath boundary, and using the boundary condition that the electric field at the boundaries is zero, gives

$$J_{ion}/J_{elec} = (m_{elec}/m_{ion})^{1/2}$$

and that

$$j_e \approx 1.85 j_o$$

where j_o refers to the one species diode. For the case of electron collection the ion current in Eq. (1) is an upper bound. If the ion current were greater, the electric field would be unable to extract all of them and a quasi-neutral pool of plasma would develop near the object. The behavior and properties of such a plasma are not within the scope of this discussion. The goal here is to define a bound such that we can be certain that such a plasma is not created.

The two species diode provides a lower bound on the maximum ion current that can be supported by a given voltage over a given distance. This is because the positive charge is in the anode-cathode gap for the maximum possible time, and the accelerating field is minimum at the anode ($E = 0$ boundary conditions are specified). Therefore, the case of ions being created throughout the diode volume can support more ion current than that allowed by Eq. (1). Indeed numerical calculations by D. L. Cooke have shown that almost an order of magnitude more ion current can be supported if it is generated uniformly throughout the

volume. The inclusion of convex geometry when considering electron collection by a spherical probe modifies this result somewhat, but for small convergence ratios the same argument holds. For larger convergence ratios almost all the path length for ion generation occurs while the convergence is small.

For convex cases, Eq. (1) can be integrated over a surface and be expressed in terms of total current.

$$\frac{I_i}{I_e} < \sqrt{\frac{m_e}{m_i}} \quad (2)$$

where the inequality comes from recognizing that Eq. (1) is an upper bound. In this form, the ion generation rate can be bounded by assuming that ionization can occur over the entire path length of a collected electron. This is an upper bound on the ion current since the threshold energy for collisional ionization is assumed to be zero. The total ion generation rate is

$$I_i < I_e \ell n_0 \sigma$$

where ℓ is the electron sheath length, n_0 is the background density of neutrals and σ is the maximum cross-section for collisional ionization by electrons.

To close this system of equations it is necessary to obtain a bound on the electron sheath length. This can be obtained by recognizing that the electron current density at the sheath boundary is less than 1.5 times the one-sided thermal flux. This result was obtained by Storey et al. for spherical and cylindrical probes. The sheath potential is bounded by beam voltage

$$V_{sh} < V_{beam}$$

Finally, since for convergent geometries the charge inside the sheath is enhanced when compared to the planar case, the planar Child-Langmuir diode formula sets an upper bound on the sheath distance

$$\ell^2 < \frac{2.32 \cdot 10^{-6} (V_{\text{beam}})^{3/2}}{1.5 j_{\text{thermal}}} \quad (4)$$

Combining Eqs. (2), (3) and then (4) a bound on the sheath neutralization can be obtained.

$$n_o < \frac{\sqrt{m_e}}{\sqrt{m_i}}, \quad (5)$$

$$n_o < \frac{\sqrt{m_e}}{\sqrt{m_i}} \frac{1}{\sigma_o \left[\frac{2.32 \times 10^{-6} V_{\text{beam}}^{3/2}}{1.5 j_{\text{th}}} \right]^{1/2}}$$

When this inequality is satisfied, ions generated within the sheath can easily be extracted by the sheath electric field. This permits the existence of a quasistatic non-neutral sheath. From the results of the double diode, the effective sheath distance for a fixed sheath potential is increased by less than 40 percent. Since the inequality was obtained as an upper bound, ionization will be a small perturbation on the electron collecting sheath, if it is satisfied.

For atomic oxygen, the peak cross-section for ionization is about 10^{-16} cm^2 and the square root of the mass ratio is $1/200$. Putting in these values for a 1 keV electron beam and a $n_e = 10^4$, = 0.1 eV plasma, sheath ionization will not be important if

$$n_o < 2 \times 10^{10} \text{ cm}^{-3}$$

or 10^{-6} torr.

This conservative bound on the maximum neutral density is satisfied for most orbiting satellites. Only through the neutral release from the satellite will this condition be violated. Numerical

studies have shown that even close to the critical density the sheath remains very nearly a Child-Langmuir diode with the sheath only fractionally larger than without ionization. In summary, sheath ionization enlarges the sheath by a small factor up to neutral densities very near (within a factor of two) the sheath breakdown density.

The inclusion of sheath ionization into the POLAR code is primarily for diagnostic purposes. The fractional ionization is computed using the tracked electron's fluxes to determine the proximity to the point where a stable sheath is not possible. Near the critical point, where the sheath size should change slightly, the plasma density within the electron attracting sheath is reduced to simulate this effect. This information is tabulated along with the total ionization within the sheath, the approximate dimensionality of the sheath (planar to spherical), and the charge reduction needed to enlarge the sheath to the proper size.

4.44.2 ANALYSIS OF THE CHARGE STABILIZED POISSON METHOD

The Charge Stabilized Poisson Method calculates for each node the maximum allowable charge that is consistent with the stability of a linearly interpolating Poisson solver. The method is developed in Section 4.44, but a further analysis is presented here to help the user interpret its impact.

POLAR's charge stabilization is accomplished through the process of charge limiting, illustrated in Figure 4.44/1. This figure shows two charge versus potential curves given by Eq. (4.44-3), rewritten here as

$$q = a \exp(-\phi_m) (\exp \phi_b - \exp \phi)$$

where $\phi_m = \ln(a\lambda_D^2/h^2)$, λ_D is the Debye length, h is the mesh spacing, and ϕ_b is the barometric potential, $\phi_b = \ln(n_i)$. For curve q_1 , $\phi_b = -3.0$ ($n_i = 0.05$); for curve q_2 , $\phi_b = 0.0$ ($n_i = 1.0$); and for both curves, $\phi_m = -2.2$, and $a = 1$. For each curve, also shown is the limiting charge given by Eq. (4.44-6) rewritten here as

$$q_L = a(\phi_b - \phi)$$

which intersects the "natural" charge curve at ϕ_c and ϕ_b . The charge stabilization method would reduce the charge to the limiting value when $\phi > \phi_c$, and use the natural charge for $\phi < \phi_c$. The parameter m provides a good measure of limiting process. From Figure 4.44/1 it can be seen that ϕ_m is the point at which the slope of the natural charge curve equals that of the limiting charge line. Figure 4.44/2 shows a family of curves giving the dependence of the cutoff potential c on the barometric potential for various values of ϕ_m . These curves were obtained by numerically solving for the zeros of the difference between q and q_L . This difference equation always has two solutions, one at ϕ_c

and one at ϕ_b , with the exception of a degeneracy at $\phi_c = \phi_b = \phi_m$, which is indicated in the figure. This figure shows that the charge limiting is minimal for $\phi_m > -1$, and quite severe for $\phi_m < -6$ or so.

Consider the first zone of a sheath to satisfy the laws of Child and Langmuir (planar space charge limiting). At the sheath edge [$z=0$] and one zone in [$z=\Delta x$] the potential and electric field are

Position	$z=0$	$z=\Delta x$
Potential	0	$K(\Delta x)^{4/3}$
Electric Field	0	$(4/3)K(\Delta x)^{1/2}$

By Gauss's Law, the charge per unit area in this zone is

$$Q/\epsilon_0 A = (4/3)K(\Delta x)^{1/3}$$

Polar computes the charge density to be $(SQALPH/(\Delta x)^2)$ times the mean potential (assuming linear interpolation), and so gets

$$Q/\epsilon_0 A = \Delta x \times (SQALPH/(\Delta x)^2) \times 1/2(0 + K(\Delta x)^{4/3}).$$

Equating these two expressions gives $SQALPH = 8/3$

Because of the economics of running a three-dimensional code, POLAR is frequently operated at high ϕ_m values, i.e., a coarse mesh with respect to the Debye length. In these cases charge is removed at almost all points. Ideally, the charge that was removed was excess charge generated by the coarse gridding. This is the artificial charge amplification argument made in Section 4.44.1. However, since POLAR must be reliably stable, the result is that too much charge is often removed. This will result in an enlarged sheath thickness for high negative ϕ_m problems. A possible cure for this would be to add a measure of charge redistribution to the stabilization algorithms.

The results of testing indicate that sheath enlargement is generally less than a zone for $\phi_m > -8$. For an α , or SQALPH = 3, this corresponds to $h/\lambda_D \approx 100$. POLAR presently makes a modest compensation for the sheath enlargement problem by placing the sheath edge at $V_S = \phi_m kT/e$. This is discussed further in Section 4.42.1.

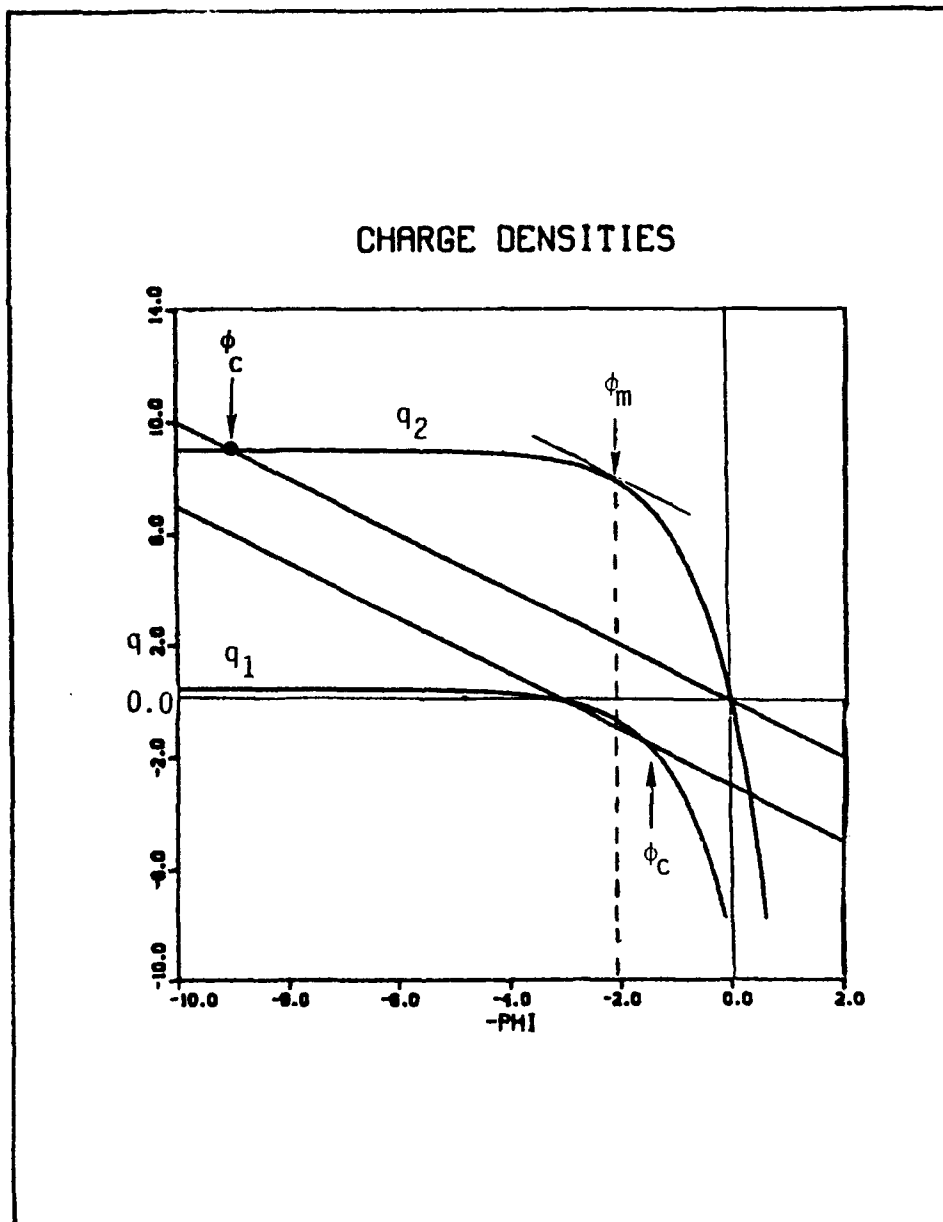


Figure 4.44/1. Plots of space charge (curves q_1 and q_2) as a function of potential as given by Eq. (4.44-3). The straight lines represent the maximum allowable charge for non-oscillatory potentials. The "natural" space charge, q_1 or q_2 , is acceptable for which slopes of the curve and the corresponding line are equal.

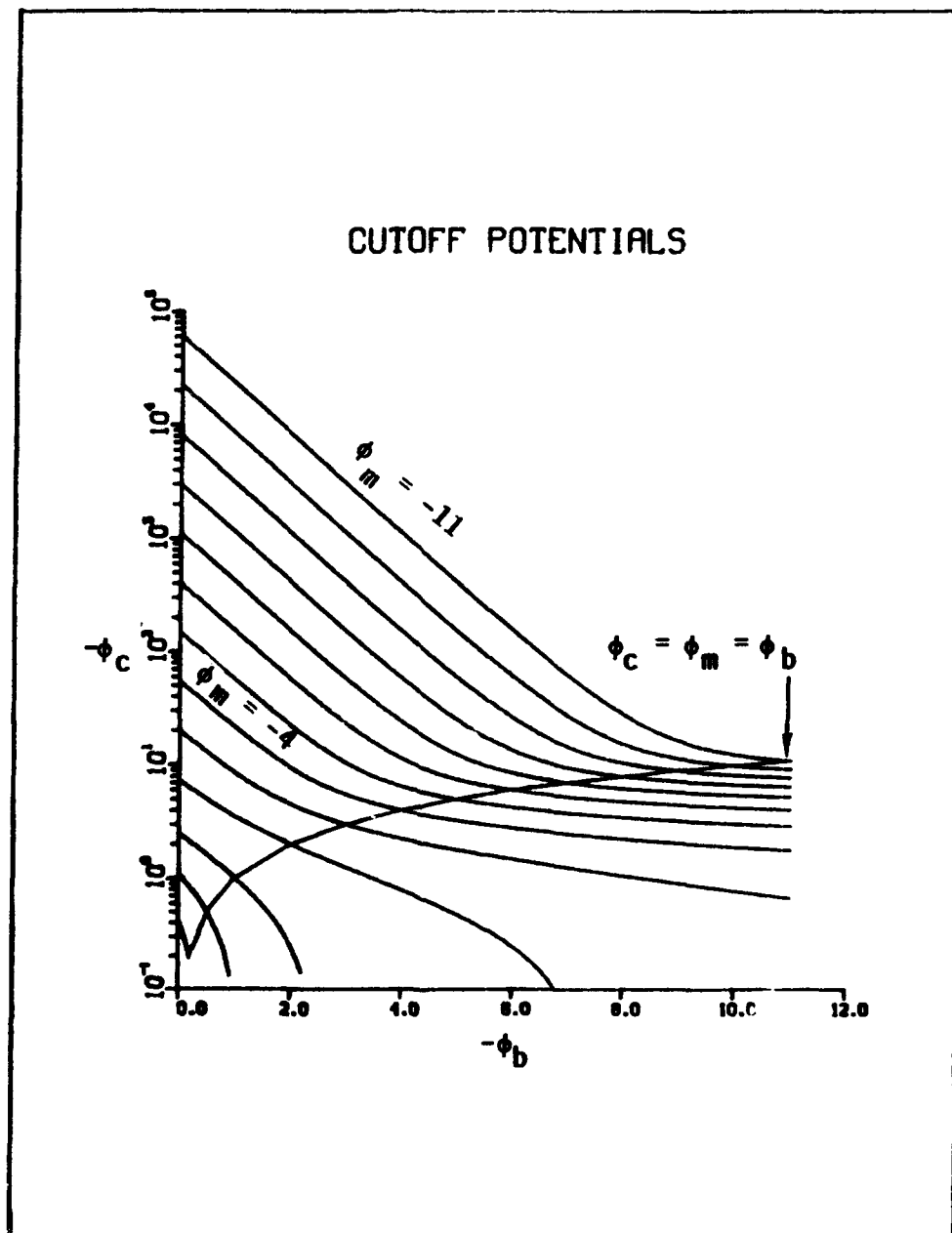


Figure 4.44/2. Plot of the space charge cutoff potential, ϕ_c , versus barometric potential ($\phi_b = \ell_n n_i$) for a series of ϕ_m values (-0.2, -0.5, -1.0, -2.0, -3.0, -4.0, ... -11.0). The point at which $\phi_m = \phi_b = \phi_c$ is also indicated.

4.44.3 PARTICLE BEAM SPACE CHARGE EFFECTS

Currently the space charge effects of particle beams are not included in POLAR.

4.44.4 ANALYTIC FORMULATION FOR SHEATH CONVERGENCE

An analytic method is available to compute space charge densities while solving Poisson's equation. This model treats space charge density as a local function of potential using a nonlinear, analytic formula appropriate to a planar or "thin" sheath. The convergence factor is computed in terms of local information and problem parameters. The Langmuir-Blodgett problem of collection by a high-voltage sphere was numerically solved and then fitted to an analytic form. An excellent fit was found with

$$(R/r)^2 = 2.29 (E\lambda_D/\theta)^{1.262} (V(r)/\theta)^{-.509} \quad (1)$$

The model also includes the effect of spacecraft motion and a magnetic field on the convergence factor. It is assumed that the routine used to describe the effect of spacecraft motion on electron current in the wake can also be used for wake ions and ram electrons and ions. The convergence factor in a magnetic field is computed using a factor f computed from the sheath distance. The sheath distance D_{sh} is computed using the minimum of Child-Langmuir distance and that computed with a Laplacian potential. f is given by

$$f = 1 - \exp(-0.5(r_L/D_{sh})^2)$$

The convergence factor is then f times that given by Equation (1) plus $1 - f$.

4.50 CHARGING MODEL

The charging equations used by POLAR were introduced in Section 3.50. In the following sections the various models employed in the solution of the charging equations are described in detail.

The general form of the charging equation that was introduced in Section 3.50 is

$$\vec{I}(t) = \vec{C} \frac{d}{dt} \vec{V}(t) + \vec{\sigma} \vec{V}(t) \quad (3.50-1)$$

where \vec{I} is the current "vector", \vec{C} the capacitance matrix, σ conductance, and V is surface voltage. Placing the equation in a differenced form, evaluating the conductance term at the advanced time (implicit) and the current at the retarded (explicit) time, it becomes

$$\vec{C} \frac{(\vec{V}(t_2) - \vec{V}(t_1))}{t_2 - t_1} + \vec{g} \vec{V}(t_2) = \vec{I}(t_1)$$

or

$$\left[\frac{\vec{C}}{t_2 - t_1} + \vec{g} \right] * (\vec{V}(t_2) - \vec{V}(t_1)) = \vec{I}(t_1) - \vec{g} \vec{V}(t_1)$$

Section 3.50 explored the difficulties associated with the above, explicit current dependence. The implicit approach replaces $I(t_1)$ with the following approximation for $I(t_2)$ for each surface:

$$I(V(t_2)) = I(V(t_1)) + \frac{dI}{dV} * (V(t_2) - V(t_1))$$

Substituting to find the implicit formulation

$$\left(\frac{C}{\Delta t_1} + g - \frac{dI}{dV} \right) \cdot [V(t_2) - V(t_1)] = I(t_1) - g V(t_1) \quad (4.50-1)$$

where $\Delta t_1 = t_2 - t_1$.

The most difficult term to evaluate in Eq. (4.50-1) is the current derivative dI/dV . A predictor-corrector type of approach is used to estimate the final voltage ($V(t_2)$) in order to calculate the derivative. Two cycles with Eq. (4.50-1) are used for each timestep iteration of the algorithm. A reasonable guess is used for the derivative in the first stage. The results from the first stage are used to find a better estimate for the final stage.

The current derivative term stabilizes the problem by increasing the diagonal matrix terms. Physically, the $\partial I(\text{surface } i)/\partial V(\text{surface } i)$ will be nonpositive for surfaces approaching a stable equilibrium. Since no surface to surface interactions are included in the current derivative matrix, dI/dV is diagonal. The current derivative is discussed in detail in Section 4.56 along with the entire charging algorithm. The current derivative for conductors is described in detail in Section 4.51.

4.51 CONDUCTOR CURRENTS AND CURRENT DERIVATIVES

With beams and sheath conduction the charging algorithm has to be applied to the exposed conductors with care. Each current source needs to be considered separately. In an attempt to describe the different situations systematically, each specific case of currents and potentials will be addressed. First, the definitions of variables to be used in the following narration.

I_b	beam current
I_p	- sheath current
I_{sec}	- secondary currents
I_{ph}	- photoelectron current
I_{raw}	- total of all currents to conductor assuming all the low energy currents (I_{sec} and I_{ph}) escape
I_{hi}	- total of all currents to the conductor excluding I_{ph} and I_{sec}
I_{cond}	- conduction current
I	- total current to conductor
dI	- total current derivative
kT	- plasma temperature
E_{sec}	- electron secondary energy, typically about 2 volts positive
V_c	- initial conductor voltage
ΔV_c	- estimated conductor voltage change made before charging calculation
V_B	- beam energy
E_{norm}	- average normal electric field over conductor
dx	- mesh or surface size

Case 1 - Conductor Voltage $\geq E_{\text{sec}}$

The currents are checked in the following order until the conductor fits a category. When the first check is satisfied, the rest are ignored. After the appropriate current check is found, the total current and the current derivative can be deduced.

$$I_{\text{hi}} > 0$$

$$\begin{aligned} \text{Then } I &= I_{\text{hi}} + I_{\text{b}} \\ dI_{\text{B}} &= \min[0, I_{\text{B}}/(V_{\text{C}} - V_{\text{B}})] \\ dI &= -(I/ΔV) + dI_{\text{B}} \end{aligned}$$

$$I_{\text{raw}} > 0$$

$$\begin{aligned} \text{Then } I &= I_{\text{raw}} \\ dI_{\text{ph}} &= 0 \\ dI_{\text{b}} &= 0 \\ dI &= -(I - I_{\text{ph}} - I_{\text{b}})/ΔV \end{aligned}$$

$$I_{\text{raw}} > 0 \text{ and } I_{\text{hi}} < 0$$

Then the total current is

$$\begin{aligned} V_{\text{C}} > E_{\text{sec}} & \text{ then } I = I_{\text{hi}} \\ V_{\text{C}} \leq E_{\text{sec}} & \text{ then } I = I_{\text{raw}} \\ & \text{and the current derivative is} \\ V_{\text{C}} \neq E_{\text{sec}} & \text{ then } dI = I/(V_{\text{C}} - E_{\text{sec}}) \\ V_{\text{C}} = E_{\text{sec}} & \text{ then } dI = I/(-kT) \end{aligned}$$

Case 2 - Conductor Voltage $< E_{\text{sec}}$

$$I_{\text{hi}} > 0$$

$$\begin{aligned} \text{Then } I &= I_{\text{raw}} - I_{\text{sec}} - I_{\text{ph}} - I_{\text{b}} \\ dI_{\text{B}} &= \min(0, (I_{\text{b}}/(V_{\text{C}} - V_{\text{B}}))) \\ dI_{\text{ph}} &= I_{\text{ph}}/(V_{\text{C}} - E_{\text{sec}}) \\ dI_{\text{sec}} &= I_{\text{sec}}/(V_{\text{C}} - E_{\text{sec}}) \\ dI &= -(I/ΔV) + dI_{\text{B}} + dI_{\text{ph}} + dI_{\text{sec}} \end{aligned}$$

$$I_{\text{raw}} < 0$$

$$\begin{aligned} \text{Then } I &= I_{\text{raw}} - I_{\text{ph}} - I_{\text{b}} \\ dI &= -(|I|/\Delta V) \end{aligned}$$

$$I_{\text{hi}} < 0 \text{ and } I_{\text{raw}} > 0$$

Then

This is the tricky case. If this conductor were an insulator, it would have a fixed normal electric field boundary condition and PWASON would find the surface potential. We will use an average normal electric field for a crude, but sufficient boundary condition. Charging is dominated by the behavior of the secondaries here.

To find the total current

$$E_{\text{norm}} \cdot dx > E_{\text{sec}}$$

$$\text{Then } I = I_{\text{hi}}$$

$$E_{\text{norm}} \cdot dx \leq E_{\text{sec}}$$

$$\text{Then } I = I_{\text{raw}}$$

and for the total current derivative

$$E_{\text{norm}} \cdot dx = E_{\text{sec}}$$

$$\text{Then } dI = I/(E_{\text{norm}} \cdot dx - E_{\text{sec}})$$

$$E_{\text{norm}} \cdot dx \neq E_{\text{sec}}$$

$$\text{Then } dI = |I|/-kT$$

The above conditions handle the complete variety of situations for exposed conductors. If the conductor is one which is not exposed at any point, then the current derivative is found by studying the derivative of the beam current. If the beam is off, then $dI = 0$. Otherwise,

$$dI = I_B/(V_C - V_B).$$

4.52 ELECTRON CURRENTS, PRIMARY, SECONDARY

The POLAR electron environment is described in Sections 3.15 and 3.31. This section explains the integration procedures used to obtain the net electron currents to a surface due to primary, secondary, and backscatter electrons when pushed electron currents are not appropriate (Section 4.52.7). Before the integrations are presented, we develop the secondary and backscatter yield coefficients. These yield models were developed for NASCAP (Reference 4-1) and are well proven and thoroughly tested. Further information concerning the code mechanics and subroutine relations can be found in Section 5.70.

4.52.1 SECONDARY ELECTRONS

Secondary electrons are defined as those emitted from the surface due to particle impact with energies below 50 eV. Their energy distribution is usually peaked below 10 eV. We define the secondary yield Y as the ratio of primary to secondary electron current.

$$Y = \frac{\text{emitted secondary current due to electron impact}}{\text{primary electron current}}$$

POLAR (NTERAK) calculates the secondary electron emission yield, Y , using the empirical formula (Reference 4-6):

$$Y(\theta) = C \int_0^R \left| \frac{dE}{dx} \right| e^{-\alpha x \cos \theta} dx$$

where x is the path length of penetration of a primary electron beam into the material, R is the "Range," or maximum penetration length, and θ is the angle of incidence of the primary electron.

This equation is based upon a simple physical model (Reference 4-7):

- a. The number of secondary electrons produced by the primary beam at a distance x is proportional to the energy loss of the beam or "stopping power" of the material, $|dE/dx|$.

- b. The fraction of the secondaries that migrate to the surface and escape decreases exponentially with depth ($f = e^{ax \cos \theta}$). Thus only those produced within a few multiples of the distance $1/a$ (the depth of escape) from the surface contribute significantly to the observed yield.

The range increases with the initial energy, E_o , of the incident electrons in a way that approximates a simple "power law" (Reference 4-8):

$$R = b E_o^n$$

where $1.0 < n < 2.0$. This equation implies a simple form for the stopping power $S(E)$:

$$S(E) = \left| \frac{dE}{dx} \right| = \left| \frac{dR}{dE_o} \right|^{-1} = \frac{E^{1-n}}{nb}$$

Because the primary beam loses energy as it passes through the material, both E , and hence $S(E_o, x)$, depend on the path length x . Integrating:

$$E^n(x) = E_o^n - \frac{x}{b}$$

$$S(x) = \frac{1}{nb} \left(\frac{b}{R - x} \right)^{1-1/n}$$

The stopping power $S(E_o, x)$ depends upon both the initial electron energy E_o , via R , and the path length x . Figure 4.52/1 shows schematically $S(E_o, x)$ plotted against x for several values of E_o . Inspection of Figure 4.52/1 and the equation for $S(x)$ illustrates the following points:

1. $S(E_o, x)$ increases with x , slowly at first, before reaching a singularity as x approaches R .
2. The initial value of $S(E_o, x)$ decreases with increasing initial energy E_o .

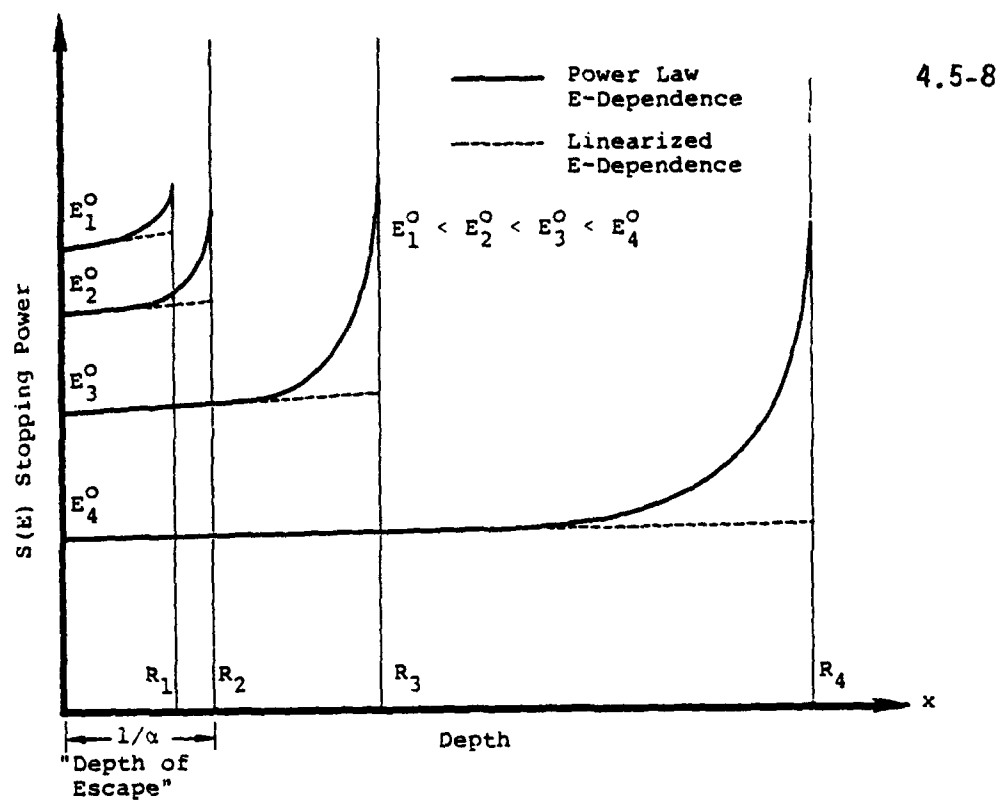


Figure 4.52/1a. Energy deposition profiles of normally incident primary electrons for incident energies E_0 .

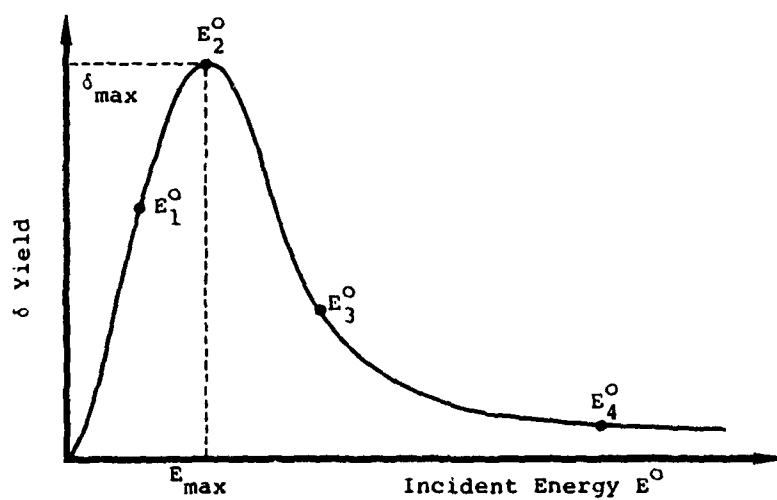


Figure 4.52/1b. Generalized yield curve.

Both of these observations are due to the decrease in electron-atom collision cross-section with increasing energy.

The yield is only sensitive to the details of the stopping-power depth-dependence for initial energies with ranges of the same order as the escape depth, $R \sim 1/\alpha$ (i.e., about the maximum of the yield curve). For lower energies, $R \ll 1/\alpha$, and essentially all of the primary energy is available for detectable secondary production, leading to a linear increase in yield with increasing E_0 . At higher energies, where $R \gg 1/\alpha$, $S(E_0, x)$ remains almost constant, at its initial value, over the depth of escape and so, along with $S(E_0, x)$ the yield decreases as E_0 increases.

POLAR takes this into account and approximates the stopping power by a linear expansion in x , about $x = 0$.

$$\frac{dE}{dx} = \left(\frac{dR}{dE_0} \right)^{-1} + \left(\frac{d^2R}{dE_0^2} \right) \left(\frac{dR}{dE_0} \right)^{-3} x$$

POLAR allows for a bi-exponential range law:

$$R = b_1 E_0^{n_1} + b_2 E_0^{n_2}$$

involving four parameters b_1, b_2, n_1, n_2 . The parameters are fit to reproduce range data as accurately as possible. For materials where no suitable data is available, a mono-exponential form is generated using Feldman's empirical relationships (Reference 4-8), connecting b and n to atomic data.

$$b = 250 A / \rho Z^{n/2}$$

$$n = 1.2 / (1 - 0.29 \log_{10} Z)$$

where A is the atomic or molecular weight of the material, Z is the atomic number, and ρ is the density. The stopping power is then

obtained indirectly via the equation above. Recently good theoretical estimates of the stopping power for number of materials have become available (Reference 4-9). Comparison of these values with those implied by the range data showed significant discrepancies, particularly for those materials fit using Feldman's formula. A better approach is to fit the four parameters in the equation for R directly to the stopping power data.

$$S = \left(n_1 b_1 E^{n_1-1} + n_2 b_2 E^{n_2-1} \right)^{-1}$$

Secondary emission of electrons due to ion impact is treated in a way similar to that for electron impact. The yield Δ is given by

$$\Delta(\theta) = C \int_0^t \left| \frac{dE}{dx} \right| e^{-\alpha x} \sin \theta \, dx$$

The angular dependence is assumed to be a simple sine form, and the stopping power is assumed to be independent of path length x over the thickness t of the sample.

$$\left| \frac{dE}{dx} \right| = \beta E^{1/2} / (1 + E/E_{\max})$$

E_{\max} is the energy at the maximum in the yield curve. This is ~50 keV for most materials. A typical yield curve is shown for aluminum in Figure 4.52/2.

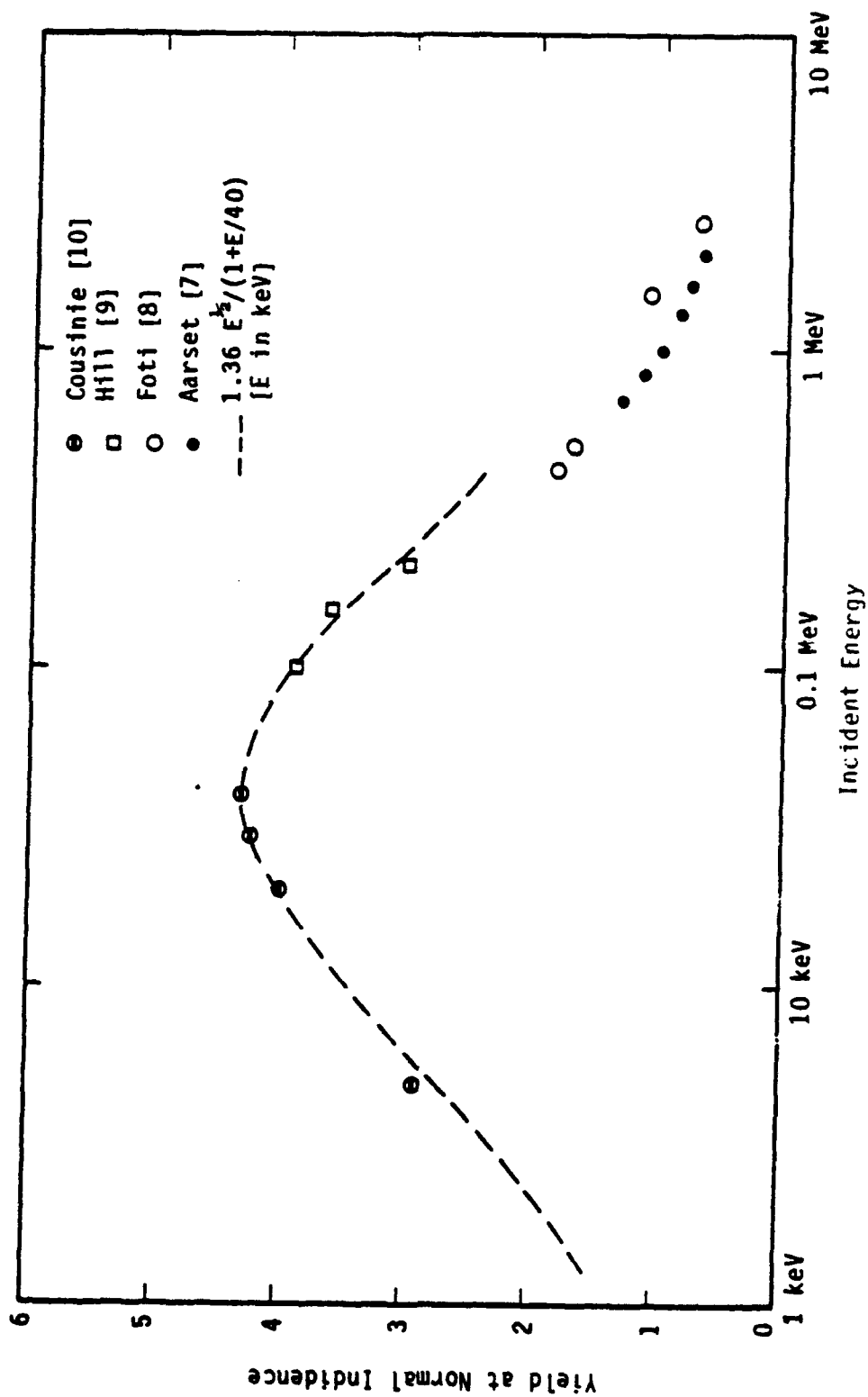


Figure 4.52/2. Secondary emission by aluminum for proton impact at normal incidence; experimental points as indicated.

4.52.2 BACKSCATTER ELECTRONS

Backscattered electrons are those emitted from the surface with energies above 50 eV. Their energy distribution is usually peaked close to the primary incident energy and they may be considered as reflected electrons.

The large-angle scattering theory, together with Monte Carlo data and experiments by Darlington and Cosslett (Reference 4-10), indicate that the angular dependence of backscattering is well described by

$$n(\theta) = \eta(0) \exp[\eta_1(1 - \cos\theta)]$$

where the value of η_1 is, within the uncertainty in the data, what would be obtained by assuming total backscattering at glancing incidence, viz. $\eta_1 = -\log \eta_0$, $n_0 = n(0)$. The net albedo for an isotropic flux is then

$$Y = 2[1 - \eta_0(1 - \log \eta_0)]/(\log \eta_0)^2 .$$

As the energy is decreased below 10 keV the backscattering increases. Data cited by Shimizu (Reference 4-11) indicated an

increase of about 0.1, almost independent of Z . POLAR approximates this component of backscattering by

$$\delta\eta_0 = 0.1 \exp[-E/5 \text{ keV}] .$$

At very low energies the backscattering coefficient becomes very small and, below 50 eV, backscattering and secondary emission are indistinguishable. POLAR takes account of this by a factor of

$$[(E - 50 \text{ eV})/\log 20] \log(E/50 \text{ eV}) .$$

The formula for energy-dependent backscattering, incorporating these assumptions, is then

$$\eta_o = \{ [\log(E/0.05)\theta(E - 0.05)\theta(1.0 - E)/\log(20) + \theta(E - 1.0) \} \times [0.1 \exp(-E/5) + 1 - (2/e)^{.037Z}]$$

where energies are measured in keV.

4.52.3 INTEGRAL OF THE MAXWELLIAN DISTRIBUTION

In Section 3.41, the current J_M of electrons of charge q to a surface at voltage V due to the Maxwellian portion of the spectrum was given as

$$J_M = \frac{qF}{(kT)^{3/2}} \int_0^{2\pi} d\phi \int_0^{\pi/2} \sin\theta \cos\theta d\theta g(E, \psi(\phi, \theta)) \\ * \int_L^\infty K dK \exp[-(qV + K)/kT]$$

$$L = \max(-qV, 0) \quad .$$

The lower kinetic energy limit L is chosen to exclude orbits that cannot energetically connect to infinity. The function g is a function of the pitch angle ψ , and total particle energy E . POLAR is presently approximating all distributions as isotropic, so the angle integrals are performed assuming g to be a constant of value unity.

The flux of electron generated secondary and backscatter electrons is obtained by adding a yield function $Y(K, \phi, \theta)$ to the above integrals (Section 3.33). POLAR presently replaces Y with an angle averaged $\bar{Y}(K)$ for the isotropic case. Thus

$$J_{MS} = \frac{qF}{(kT)^2 \pi} e^{-qV/kT} \int_L^\infty K e^{-K/kT} \cdot \bar{Y}(K) \cdot dK$$

where the primary current J , is obtained by setting $\bar{Y} = 1$.

For arbitrary \bar{Y} , the integral must be performed numerically. It is also desirable to divide up the spectrum in a manner that covers the larger fluxes at low energies without ignoring the high energy tail of the Maxwellian spectrum. A logarithmic spacing is accomplished by the substitution

$$K = -kT \ln(x)$$

Thus we have

$$J_{MS} = qF \int_{x_l}^{x_u} \bar{Y}(K(x)) \ln(x) dx$$

where $x_u = e^{-L/kT}$, and $x_l = 0$. Since $\ln(x)$ is singular at $x = 0$, the lower limit is set to $x_l = 0.01 * x_u$ and the omitted portion of the spectrum approximated by $x_l * \ln(x)$. The summation is performed using Simpson's rule and 20 points.

4.52.4 INTEGRAL OF THE POWER LAW DISTRIBUTION

The discussions of angular dependence in the flux integral given in Section 3.31 and 4.52.3 also apply here, so we will limit this treatment to the energy integral

$$(A) \quad J = aq \int_{KL}^{KU} \bar{Y}(k) \cdot (J + qV)^{-(a+1)} K \, dK$$

where J is current, \bar{Y} is secondary or backscatter yield, K is kinetic energy, a is a constant ($a = \pi A$ for isotropy), q is charge, and V is surface potential. For the limits we choose

$$KL = \text{MAX} (0, EL - qV)$$

$$KU = \text{MAX} (EU, EU - qV)$$

where EL is a physical cutoff (default = 100 eV) and EU is imposed sufficiently high as to avoid significant error (default = 1×10^9 eV).

The first step is the transformation

$$X = K + qV ,$$

which gives

$$J = aq \int_{XL}^{XU} \bar{Y}(K(X)) \cdot [X^{-a} - qV X^{-(a+1)}] \, dX$$

where $XL = \text{max}(qV, EL)$ and $XU = \text{max}(EU + qV, EU)$.

Since a can be as large as 3.0, this spectrum is strongly peaked towards lower energies, which implies that a nonuniform spacing of integration points is desirable. This is easily accomplished by the substitutions,

$$x = y^{-1/(a-1)}, \text{ for the first term, and}$$

$$x = z^{-1/a}, \text{ for the second term}$$

of the previous integral, which leads directly to

$$J = aq \left[\frac{1}{a-1} \int_{y_l}^{y_u} \bar{Y} dy + \frac{1}{a} \int_{z_l}^{z_u} \bar{Y} dz \right] .$$

where $y_l = x_l^{1-a}$, etc. Notice that these choices produce integration weights of value unity. The numerical integration is done over 20 points spaced evenly in y and z . These transformations are very strongly biased towards the lower energies thus the upper cutoff was chosen quite high (1×10^9 eV) to force good coverage of intermediate energies where \bar{Y} might be peaked.

A method with variable bias was also investigated. This utilized for Eq. (A), the substitution

$$K = x^{-1/\beta} - S$$

$$dK = -\frac{1}{\beta} x^{-(1+1/\beta)} dx = w(x) dx .$$

This method has the ability to adjust the bias with β , and center the integration points about an energy related to S . This method is inherently slower due to the calculation of the weights $w(x)$, and appeared to offer no great advantage over the previous method. It is not presently used by POLAR, but remains an option.

4.52.5 INTEGRATION OF GAUSSIAN DISTRIBUTION ELECTRONS

This treatment is limited to the energy integral of the Gaussian electron distribution. The angular dependence has been integrated assuming isotropy as discussed in Section 3.41 and 4.52.3. From Section 3.41, the integral of interest is

$$J_S = \pi B \int_{KL}^{\infty} \bar{Y}(K) \exp[-(K-K_0)^2/\delta^2] k \, dK . \quad (1)$$

where $KL = \max(0, -qV)$.

As in the previous Section 4.52.1 and 4.52.2, the inclusion of the angle averaged yield function $\bar{Y}(K)$ will make J the current of backscatter or secondary electrons (see Section 3.43); with the emission of \bar{Y} , J becomes the incoming primary flux.

This integral is performed numerically by an 8 point Hermit integration scheme (Reference 4-12).

The weights and abscissa are:

i	x_i	$w(x_i)$
—	—	—
1,5	± 0.38119	6.61147×10^{-1}
2,6	± 1.15719	2.07802×10^{-1}
3,7	± 1.98166	1.70780×10^{-2}
4,8	± 2.93064	1.99604×10^{-4}

Equation (1) can thus be approximated by

$$J_S = \pi B \sum_{i=1}^8 w(x_i) \cdot \bar{Y}(K_i) \cdot \exp[-\Delta K_i^2/\delta^2] \cdot K_i \cdot \theta(K_i - KL) \quad (2)$$

where

$$\begin{aligned} K_i &= K_0 + X_i \cdot \delta \\ \Delta K_i &= X_i \cdot \delta \end{aligned}$$

and

$$\theta(K_i - K) \begin{cases} = 1 & \text{for } K_i - KL > 0 \\ = 0 & \text{for } K_i - KL < 0 \end{cases}$$

KL is chosen as $\max(0, -qV)$ which excludes energetically trapped orbits.

J_S as given by Eq. (2) has the undesirable property of being discontinuous with respect to K due to the θ function. This feature can be removed by calculating J with the same scheme (omitting \bar{Y}). J will be discontinuous at exactly the same values of K as J_S . We can then form a smooth J_S^* as

$$J_S^* = \frac{J_S}{J} \cdot J_a$$

where J_a is the analytic solution to the primary flux integral derived below:

$$J_a = qB \int_{KL}^{\infty} \exp \left[-\left(\frac{K - K_0}{\delta} \right)^2 \right] K \, dK$$

set

$$X = (K - K_o)/\delta, K = X\delta + K_o$$

so

$$J_a = \pi B \int_Z^{\infty} \exp(-X^2) (X\delta + K_o) dX$$

where

$$Z = \frac{K_L - K_o}{\delta}.$$

Integrating

$$J_a = \frac{\pi B \delta^2}{2} \left[\exp(-Z^2) + \frac{\sqrt{\pi}}{\delta} \left[(K_o + |K_o| \operatorname{erf}(|Z|)) \right] \right]$$

4.52.6 PHOTOEMISSION

The photoemission electron currents are calculated using material properties and the area of the surface lit by the sun. The material property (6.12.10) used is the yield (Y), or the number of electrons emitted for a surface normal exposed to the solar spectrum, an "earth distance" from the sun. POLAR calculates the photocurrent from a surface exposed to the sun at an angle ϕ , according to

$$i_{\text{phot}} = (\text{Area exposed}) \cdot Y \cdot \cos\phi$$

where the exposed area takes into account any shadowing by other surfaces of the object. This formula assumes that the yield per photon is, on the average, independent of ϕ .

4.52.7 SHEATH TO OBJECT ELECTRON CURRENTS

When the attracted species are electrons, the initial currents and velocities are different than for ions. This discussion applies to attracted electrons whose collection is space charge limited; the orbit limited case follows. To account for sheath shadowing in the quasineutral presheath the sheath flux is

$$J_{\text{sheath}} = Ne \sqrt{kT/2\pi m}$$

where N is the ion density in the presheath. For a nonflowing plasma $N = N_0$, the unperturbed density; however, for a flowing plasma in the wake direction, N is reduced over the ambient value, by geometric shadowing effects of the sheath. (Section 3.31.)

The initial velocity for electrons used in the particle pusher routines is average velocity of an electron crossing the sheath boundary. This is

$$V_0 = (\sqrt{2 kT/\pi m}) \hat{E}$$

The unit vector \hat{E} is opposite to the direction of the electric field at the sheath boundary.

The electron tracking is broken into two categories. When the cyclotron radius is large (greater than a mesh size) step pushing is used as for ions. If the cyclotron radius is smaller than a mesh size, a drift approximation is used (the actual keyword/variable used is RDMAX2). The cyclotron radius is found by

$$r_{\text{cyc}} = m_e v' / eB$$

where \vec{v}' is the velocity in the drift frame. The drift frame is the frame in which the perpendicular electric field is zero. The drift approximation ignores electric field gradients and that is consistent with the finite element approximation. The drift velocity is given by

$$\vec{v}_D = \frac{\vec{E} \times \vec{B}}{B^2} .$$

Writing the initial velocity and electric field in components parallel to B ($E_{||}$ and $V_{||}$) and normal to B (E_{\perp} and V_{\perp}) the drift approximation is

$$m \frac{dV_{||}}{dt} = -e E_{||}$$

and

$$\vec{V}_{\perp} = \frac{\vec{E} \times \vec{B}}{B^2} .$$

The procession $\vec{V}' = \vec{V} - \vec{V}_{\perp}$ about B is computed in the drift frame when

$$\frac{d\vec{V}'}{dt} = \vec{\omega} \times \vec{V}'$$

when

$$\vec{\omega} = \frac{e\vec{B}}{m} .$$

In orbit limited electron collection, the surface fluxes are defined by

$$J_{\text{surface}} = J_{\text{sheath}} (1 + V_{\text{surface}}/kT)$$

For more information, see the NASCAP Programmer's Reference Manual (Ref. 4-1).

4.53 ION SURFACE CURRENTS

The ion currents used in POLAR are found using one of two methods. When surfaces have potentials which are not high enough to be enclosed by the sheath, the random ion current (Section 4.53.10) is used. These currents are also used as default currents when the CURREN module has not been utilized previously during the run (for an example see PRECHG in Section 6.42.40).

The second method uses the particles pushed by the CURREN module (Sections 4.40 and 5.60) to the object's surface. These pushed currents are combined with the random ion currents to provide ion currents with a voltage dependence. Section 4.53.20 describes the second method.

The ion current contribution to the current derivative (dI/dV in Eq. (4.50-1) is discussed in Section 4.53.30.

The above pertains to attracted ion currents whose collection is space charge limited. For a discussion of orbit limited ion currents, please see the NASCAP Programmer's Reference Manual (Reference 4-1).

4.53.10 THERMAL ION SURFACE CURRENTS

POLAR must have a default model for ion currents for surfaces that are not included inside the sheath, or have not yet had an ion current calculated.

For an uncharged surface in a non-flowing plasma, its "random" thermal ion current density would be

$$J_{th} = N_o e \sqrt{\frac{kT}{2\pi m_i}}$$

For the case of a flowing plasma the current should be properly derived by integrals over the ion distribution function. We have thus far found it adequate to approximate the current by blending the following approximation. For a surface facing the flow

$$J \approx N_o e (-\vec{V}_n \cdot \vec{a}) \sqrt{\frac{kT}{m_i}}$$

where \vec{a} is the surface normal, and \vec{V}_n is the flow or Mach velocity, normalized to the ion acoustic speed

$$V_c = \sqrt{\frac{kT}{m_i}}$$

For a surface on the wake side, we presume that a neutral ion density, n_g , has been calculated, and approximate that

$$J \approx n_g \cdot J_{th}$$

These approximations are combined into the expression

$$J_i = J_{th} \cdot n_g \left[\sqrt{2\pi} \cdot VDN + A \right]$$

Where

$$VDN = \max(0.0, -\vec{V}_M \cdot \hat{a})$$

$$A = \min(2.0, n_g^{-1})$$

4.53.20 SHEATH TO OBJECT ION CURRENTS

For the space charge collection limited, attracted ion currents, POLAR tracks particles from a sheath surface to an object surface. Experience has shown that if surface currents are simply accumulated from incident particles, the currents are noisy and lead to non-physical charging behavior. Numerous reasons exist for this noise: tracking errors, potential field irregularities, too few trajectories, etc. While these problems have all been studied, it remains desirable to have a smoothing algorithm for the ion surface currents.

The ion currents to be smoothed are derived from the "dead-list" of particles produced from the particle pushing module CURREN (4.40). This dead-list contains the particle's weight, final position, final velocity, and initial energy and the surface the particle landed on. To speed up the surface current calculation and to reduce noise, the dead-list is condensed to the SRFC list, ordered by surface number. The particle weights are added since they represent the ion current. The particle weights are also used to weight the averages of spatial and energy information. An average position on the cell, angle of incidence, and particle energy are calculated as follows:

$$\bar{X}_i = \frac{\sum_{k=1}^N w_k X_{ik}}{\sum_{k=1}^N w_k}$$

where w_k is the weight (current contribution) of particle k , N is the number of particles striking the surface, and X_i is the i^{th} component of the position vector. The average velocity and energy are found similarly.

The adopted smoothing algorithm is a two-step process wherein the raw surface currents are distributed to nodes, and then re-distributed back to surfaces. This simple algorithm is illustrated in Figure 4.53/1. Given an averaged particle at the point P, on surface a, its current is shared in a bilinear fashion to the vertices (nodes) of the surface, producing a node current T.

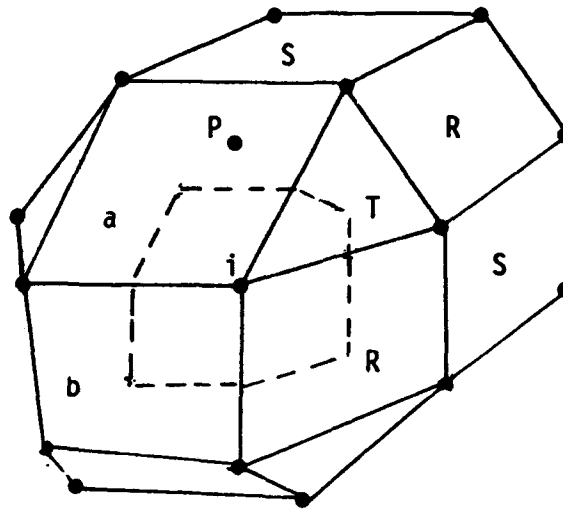


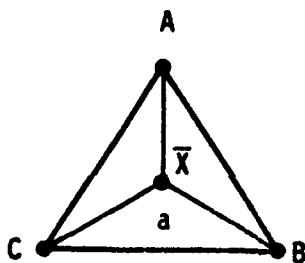
Figure 4.53/1.

Since two different types of surface occur, triangles and rectangles, two different methods are used. For triangles, the bilinear weight of a corner is area of the triangle opposite the corner over the total surface area (see Figure 4.53/2).

$$wb_{sa} = \frac{\text{area } a}{\text{area of triangle } ABC}$$

or

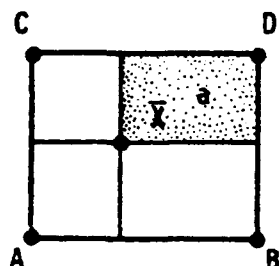
$$wb_{sa} = \frac{|(\vec{X}-\vec{C}) \times (\vec{B}-\vec{C})|}{|(\vec{A}-\vec{C}) \times (\vec{B}-\vec{C})|}$$



where w_{sn} is the bilinear weight, \bar{X} is the particle position, and s and a refer to surface and node.

Figure 4.53/2. Bilinear weighting of triangular surfaces.

For rectangular surfaces, the particle position divides the rectangle into four smaller rectangles. Then the weight is found by dividing the area of the opposite small rectangle by the area of the surface (see Figure 4.53/3).



$$wb_{sa} = \frac{\text{area of } a}{\text{area of } ABCD}$$

Figure 4.53/3. Bilinear weight (wb_{sa}) of a rectangular surface.

Thus we have for Figure 4.53/1

$$\Delta I_i = wb_{ai} \cdot \text{SRFC}(a)$$

where the Δ indicates that this is a current increment. The complete I_i is never formed.

The next step is to share the node current back to the surfaces. We may derive a simple sharing formula by forming a nodal current density, ΔJ_i , as

$$\Delta J_i = \Delta I_i / A_i$$

where A_i is the area associated with node i ;

$$A_i = \sum_{j=1}^{m_i} A_j / n_j$$

where the A_j are areas of the m surface cells adjoining node i , and n_j is the number of vertices of each adjoining surface cell. A_i is bounded by the dashed line in Figure 4.53/1.

The ΔJ_i is redistributed to surfaces adjoining node i in proportion to the area each contributed to the node area. Thus, surface b would receive a final surface current increment ΔI_b ,

$$\Delta I_b = \frac{A_b}{n_b} \Delta J_i$$

Finally, we may combine the three previous equations into a node to surface weight factor ws_{ij} . Thus the final surface current, ΔI_b , is obtained from the intermediate node current ΔI_i as

$$\Delta I_b = ws_{ib} \Delta I_i$$

$$ws_{ib} = \frac{A_b n_b}{\sum_{j=1}^{m_i} A_j / n_j}$$

This process is performed for each surface adjoining the i nodes of surface a . The final smoothed surface currents are accumulated as this overall process is repeated for all the surfaces of the object.

An important feature of this algorithm is that a uniform flux to an irregular object will produce surface currents exactly proportional to the surface areas as would be expected. To see this, consider the quasisphere of Figure 4.53/1. This object has three types of surfaces, with areas S , R , and T ; and only one type of node. A uniform flux of particles, if tracked accurately, will produce uniform node currents. Summing $w_{ij} * I_i$ around the vertices of a surface, we see that the resulting surface current will be proportional to the surface area, and inversely proportional to the factor in the denominator of the expression for ws_{ij} , which is constant for our example. The ws_{ij} are properly normalized. This can be easily seen in our example by summing the ws_{ij} around a node;

$$\sum_{k=1}^{n_i} \frac{A_k/n_k}{\left(\sum_{j=1}^{n_i} A_j/n_j \right)} = 1.0$$

since the indices k and j range over the same surfaces.

At our present stage of development, we are simply dividing and redistributing the node currents according to the relative areas of adjacent cell, but provisions have been made for a more comprehensive treatment. For example, spatial and electrical information could be used to avoid non-physical sharing of surface currents; such as redistributing ion currents to a surface with a large positive potential, or around corners.

After the sheath to object ion currents have been calculated, they need to be combined with the ambient or random ion currents (see Section 4.53.10) since some surfaces may not be within the sheath. The two sets of currents are blended by taking the maximum of the two as the surface current.

4.53.30 ION CURRENT DERIVATIVE

In order to increase the utility of the ion currents in the charging algorithm, an approximate voltage dependence needs to be defined. The algorithm used by NTERAK chooses a model depending on current voltage and the last voltage change (or estimated voltage change).

The derivative is calculated to estimate $I(V_1)$, where

$$I(V_1) = I(V_0) + \frac{dI}{dV} \Delta V$$

For V_1 less than $-10 kT$, an ion attracting surface,

$$\frac{dI}{dV} \Delta V = - \frac{3I(V_0) (V_1 - V_0)}{2(|V_0| + DVLIM)}$$

where DVLIM is used to limit the voltage change in a charging single step. This is a Child's law ($I \approx -|V|^{-3/2}$) dependence which compensates for overcorrection of the voltage dependence for small sheaths.

When V_1 is greater than $-10 kT$ and the surface is charging positively ($V_1 - V_0 > 0$), then the current derivative is

$$\frac{dI}{dV} \Delta V = - \frac{I(V_0) (kT - V_0)}{(|V_0| + kT)}$$

and if it is charging negative,

$$\frac{dI}{dV} \Delta V = 0 \quad .$$

These ion current derivatives have been found to produce stable results when only one CHARGE iteration is performed between PWASON/CURRENT CYCLES. It is recommended only one CHARGE iteration be done without recalculating space potentials and surface ion currents when the current balance on a surface is affected more by ion currents than by electron secondaries. When secondaries are dominant, the ion current derivative becomes insignificant and multiple CHARGE iterations are possible and recommended.

The value of I_1 is limited in later portions of the surface charging calculation (Section 4.56.20). The total current due to all current sources is constrained in order to force the problem to smoothly converge on a solution. This allows the slower portions of the calculation, PWASON and CURRENT, to execute more quickly. The constraints discussed in Section 4.56.20 may result in the re-evaluation of I_1 at a different V_1 .

4.54 SURFACE INTERACTIONS

Surfaces interact with each other via conduction, electrical or photocurrents. Insulating surfaces interact with other insulating surfaces by surface conduction and their underlying conductor by bulk conductivity (Section 4.54.10). When normal electric field boundary conditions are in effect, a hopping current carries low energy electrons to the most positive conductor in the area. For the purposes of discussion, this current is called photoconduction although it may be caused by phenomena other than photons (such as positively biased, exposed conductors) and is described in Section 4.54.20.

Surface to plasma and to conductor capacitances are discussed in 4.54.30 and 4.54.40, respectively.

4.54.10 SURFACE CONDUCTIVITY

POLAR breaks the surface of the satellite into smaller surfaces using the grid, so that each portion of the surface is completely contained in a single element. Because these surfaces are not really disjoint, the surface to surface conduction current is taken into account when both surfaces are insulators. Normally the current due to surface conductance is several orders of magnitude less than the bulk conductance, the conductance from the insulator surface to the underlying conductor.

The bulk conductance is found by using the material properties (Section 6.12.10) defining σ_B , bulk conductivity ($\Omega^{-1} \text{ m}^{-1}$, property 3) and d , the dielectric thickness (m, property 2). So

$$\text{Bulk Conductivity} = \frac{A\sigma_B}{\epsilon_0 d}$$

where A is the surface area in m^2 and ϵ_0 is permittivity of free space (8.85×10^{-12} farad/m).

The surface to surface conductivity is calculated by assuming each surface is a square then using the surface resistivity. Or for a surface i ,

$$R_i = \frac{1}{2} \rho_i$$

Where ρ is the surface resistivity ($\Omega \text{ edge}^{-1}$, property 14). The surface conductivity along the edge between surface i and j would be

$$\text{Surface Conductivity} = \frac{1}{R_i + R_j}$$

This approximation is accurate enough since surface charging is dominated by other effects.

4.54.20 PHOTOCONDUCTION/HOPPING SECONDARIES

An electron current through space above a surface can exist when secondaries are emitted but trapped by a positive surface potential. These secondaries can be generated by photons or by high electron fluxes inside electron collecting sheaths. The hopping secondary electron currents appear to be highly voltage sensitive surface currents. The currents move along parallel electric fields until they are absorbed by a conductor.

The presence of the parallel electric and hopping current informs CHARGE it will not be able to calculate surface potentials for these surfaces. So it fixes their surface potentials. But their contribution to the current of the exposed conductor where they end up must be considered.

The current to the exposed conductor can be found by

$$I = \int_{\text{conductor}} \nabla \cdot \mathbf{J} \, dA$$

$$I = \int_{\text{boundary of conductor}} \mathbf{J} \cdot d\mathbf{l}$$

$$I = \int_{\text{boundary}} - \frac{4\langle\epsilon\rangle}{E_{\perp}^2} J_e E_{||} d\mathbf{l}$$

where J_e is the incident electron current (see Section 4.22 for more detail).

This is done in the code by adding the incident current to insulating surfaces (times a multiplying factor) adjacent to an exposed conductor to the exposed conductor's current. In this manner, photoconduction, or hopping secondary currents, are modeled.

4.54.30 SURFACE TO PLASMA CAPACITANCE

The small surface to plasma capacitance is currently modeled using

$$C_{sm} = \frac{\epsilon_0 A}{h}$$

where ϵ_0 is the permittivity of free space, A is the area of the surface and h is the mesh length. This approximation appears to be accurate enough for modeling purposes.

4.54.40 SURFACE TO CONDUCTOR CAPACITANCE

The surface to conductor capacitance determines the charging or differential charging rates for many problems. NTERAK calculates the capacitance using the following material properties (Section 6.12.10): relative dielectric constant (ϵ_r , property 1) and dielectric

thickness (d, property 2). So the large capacitance in charging problems is

$$C_{lg} = \frac{\epsilon_r \epsilon_o A}{d}$$

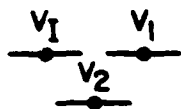
where A is the surface area.

4.55 CIRCUIT MODEL

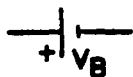
POLAR represents the spacecraft in the charging algorithm with a circuit model using the plasma as a voltage dependent current source. Insulating surfaces are represented by circuits of the form shown in Figure 4.55/2. (Figure 4.55/1 defines the circuit elements used in Figures 4.55/2-4.) Exposed conductors follow the example of Figure 4.55/3. Additional connections appear between surfaces and between conductors. Insulating surfaces are connected as shown in Figure 4.55/4. Underlying conductors are connected to each other via a resistor and a capacitor in parallel. The exposed conductor surfaces are lumped together for each of the underlying conductors. An example of a general circuit and its charging equations has been worked out in detail in Section 4.57.



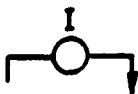
plasma ground

voltage on insulating surface
(surface 1, surface 2)

underlying conductor reference voltage



voltage due to magnetic field



current source, from plasma to surface



resistance (1/conductance)

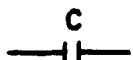
capacitor (those with a subscript G
are plasma to surface capacitances)

Figure 4.55/1. Legend of circuit elements shown in Figures 4.55/2 - 4.55/4.

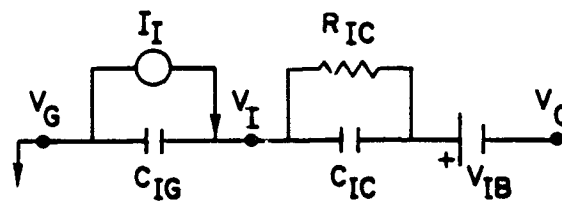


Figure 4.55/2. Circuit representation of a single insulating surface (see Figure 4.55/1 for a legend of the circuit elements).

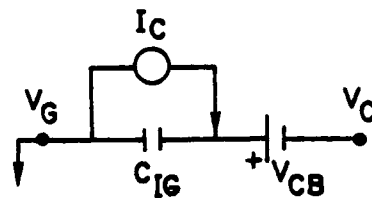


Figure 4.55/3. Circuit representation of a single exposed conductor surface (see Figure 4.55/1 for a legend of the circuit elements).

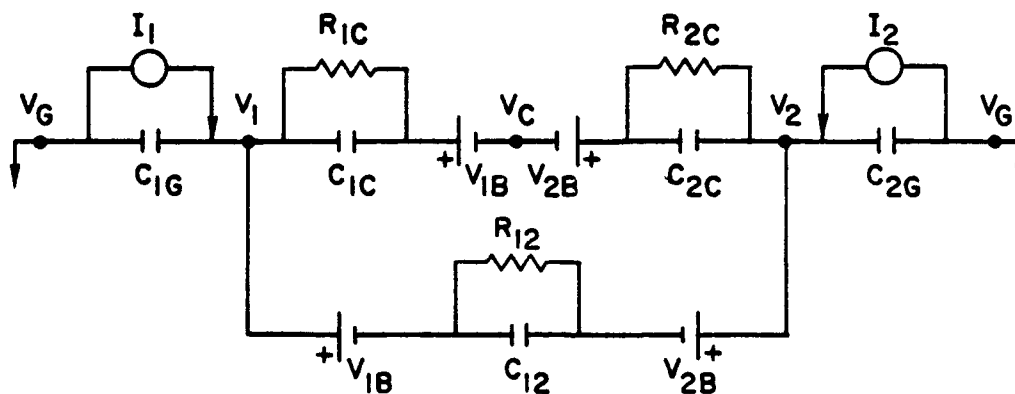


Figure 4.55/4. Circuit representation of two insulating surfaces with a common underlying conductor (see Figure 4.55/1 for a legend of the circuit elements).

4.56 CHARGING ALGORITHM

This section discusses the details of the charging algorithm. Section 5.7 describes the actual coding used for the surface charging module. Here the physical models are presented in an order which parallels the sequence followed by the coding.

4.56.10 CHARGING NOTATION

The two stage iteration over the implicit charging equation (4.50-1) requires the careful use of a notation to prevent confusion. Subscripts shall be used to indicate timestep iterations of the variables. A superscript prime marks a result found during the first stage. And a superscript double prime indicates the voltage limited result from the first stage. So the first stage, first iteration charging equation would be

$$\left[\frac{\tilde{C}}{t_1 - t_0} + \tilde{g} - \frac{d\tilde{I}}{d\tilde{V}} \bigg|_{t_0} \right] (\tilde{V}'(t_1) - \tilde{V}(t_0)) = \tilde{I}(t_0) - \tilde{g} \tilde{V}(t_0)$$

or perhaps more simply,

$$\left[\frac{\tilde{C}}{\Delta t_0} + \tilde{g} - \frac{d\tilde{I}}{d\tilde{V}} \bigg|_0 \right] (\tilde{V}'_1 - \tilde{V}_0) = \tilde{I}_0 - \tilde{g} \tilde{V}_0$$

Note the convention of using (t_0) to define the initial values.

The second stage, first iteration equation which finds \tilde{V}_1 (the initial voltage for the next iteration) would be

$$\left(\frac{C}{\Delta t_0} + g - \frac{dI}{dV} \bigg|_0 \right) (V_1 - V_0) = I_0 - g V_0$$

Generalizing for i^{th} iteration, the two stages become,

$$\left(\frac{C}{\Delta t_i} + g - \frac{dI}{dV} \bigg|_i \right) (V'_{i+1} - V_i) = I_i - g V_i \quad (4.56-1)$$

$$\left(\frac{C}{\Delta t_i} + g - \frac{dI}{dV} \bigg|_i \right) (V_{i+1} - V_i) = I_i - g V_i \quad (4.56-2)$$

where dI/dV has yet to be defined.

For the first stage, the current derivative is estimated using the initial values described in the next section. The derivative used by the second stage is found by using the results of the first stage. It is defined by

$$\frac{dI}{dV} \bigg|_i = \frac{I'_{i+1} - I_i}{V'_{i+1} - V_i} \quad (4.56-3)$$

where I'_{i+1} is the surface current evaluated using V'_{i+1} , the voltage limited surface voltages from the first stage, or

$$V'_{i+1} = f(V_{i+1})$$

Although there are two stages for each iteration, only the current derivative changes from stage to stage. Eqs. (4.56-1) and (4.56-2) can be simplified somewhat more to

$$\tilde{M}_i \Delta Y_i' = R_i \quad (4.56-4)$$

for the first stage and the second stage equation,

$$\tilde{M}_i' \Delta Y_i = R_i \quad (4.56-5)$$

where

$$\tilde{M}_i = A_i - \left. \frac{dI}{dY} \right|_i$$

$$\tilde{M}_i' = A_i' - \left. \frac{dI}{dY} \right|_i'$$

$$A_i = \frac{\tilde{C}}{\Delta t_i} + g$$

$$\Delta Y_i' = Y_{i+1}' - Y_i$$

$$\Delta Y_i = Y_{i+1} - Y_i$$

$$R_i = I_i - g Y_i$$

The above abbreviated notation will prove useful during general algorithm and code discussions.

4.56.20 DETAILS OF THE CHARGING ALGORITHM

The formulations of the charging equations defined in Section 4.56.10 will be used to describe the quantities used by NTERAK to model surface charging.

The conductance matrix, \underline{g} , contains the bulk, surface to surface, and conductor to conductor conductivity terms. The capacitance matrix, \underline{C} , is constructed from the interconductor, bulk and surface to infinity capacitance terms. Currently the capacitance from the surface to infinity uses a distance to the sheath of 1 meter.

The total surface currents, \underline{I} , are calculated using the surface voltages, \underline{V} . The models for the ion and electron currents to the surface have been discussed in Sections 4.53 and 4.52, respectively.

The voltage change during a stage, $\Delta \underline{V}_i'$ and $\Delta \underline{V}_i$, is found by solving Eqs. (4.56-4) and (4.56-5), respectively, using the Incomplete Cholesky Conjugate Gradient (ICCG) method described in Section 4.30. The ICCG matrix solver solves the charging equations quickly and efficiently even for large satellites.

The crucial portion of the implicit charging equation is the current derivative, $d\underline{I}/d\underline{V}$. Each of two stages has a different method of calculating the derivative.

The first stage attempts to make an intelligent guess based on user defined constants and results from previous iterations. Two possible derivatives may be calculated for each surface. The final matrix, which is diagonal, is built of these independent terms. The first defines the crossover or equilibrium voltage, where the total current is zero, to be $XDVFAC \cdot DVLIM$ volts away ($XDVFAC$ and $DVLIM$ are input parameters discussed in Section 6.43.30). Or as it is computed on a surface by surface basis using

$$\left(\frac{dI}{dV} \right)_1 = - \frac{|I_i|}{XDVFAC * DVLIM}$$

The other calculation method is used when previous iteration information indicates the crossover voltage has been overshoot. Obviously this method cannot be used for the first iteration. The second method is

$$\left(\frac{dI}{dV} \right)_2 = - \left| \frac{2 * I_i}{V_i - V_{i-1}} \right|$$

which predicts equilibrium to be reachable with half the voltage change of the previous timestep iteration. When both forms are applicable (i.e., after an overshoot by a surface in at least the second iteration), the more negative current derivative is used.

The voltage change, $\Delta V'_{i,}$, found by ICCG using the first stage current derivative can then be solved to find the new predicted surface voltage list, V'_{i+1} . This result is used to find the second stage current derivative after a voltage limiting process.

Finding the most useful voltage, V''_{i+1} , for the second implicit stage current derivative requires two steps. First, the voltage change estimated during the first stage may need to be modified to correct obvious problems, like voltage overshooting or destabilizing voltage changes due to the influence of other surfaces.

Figure 4.56/1 shows a hypothetical I-V curve possessing multiple crossovers. Upper and lower boundaries enclose the crossover voltages and are used to limit the initial voltage approximation. This keeps the surface voltage from straying too far from the appropriate

crossover. This feature also stabilizes the surface charging by reducing the voltage changes in regions where the I-V curve has a rapidly varying slope. It also allows the user to exercise his knowledge to produce a more efficient charging sequence by choosing the boundary voltages. If a surface voltage is outside the bounded region and moving away from the nearest equilibrium point, the estimated voltage change is limited to a maximum of DVLIM (Section 6.43.30). Since the potential is leaving the vicinity of the crossover point, slowing the movement helps to stabilize the problem. Usually this event is due to the influence of another capacitively coupled surface.

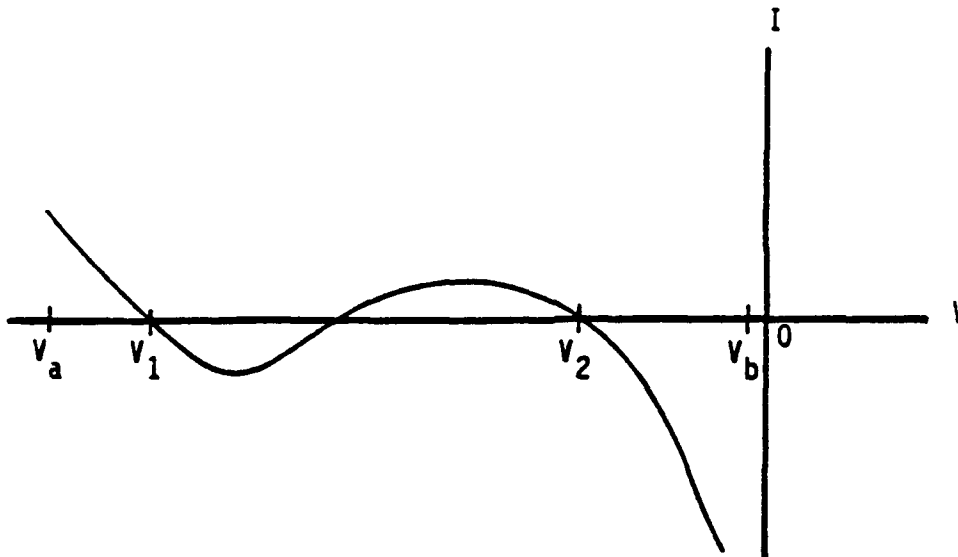


Figure 4.56/1. A multiple crossover I-V curve. Voltages V_a and V_b mark the upper and lower boundaries of the region known to enclose both the stable equilibrium points, V_1 and V_2 .

Another problem caused by a second surface is a surface potential being driven in the direction opposite to the one expected due to the environmental fluxes. It is presumed that during the second stage the external influence (usually the underlying conductor) will be constrained and greatly reduced. Thus, the sign of the voltage increment is changed to counteract this effect to establish a more reasonable dI/dV .

The next step is to calculate a current for the voltage found above. If the new current has the same sign as the initial current, the crossover has not been reached yet and a derivative can be calculated using the two points. If the sign of the current has changed, the surface potential overshoot the crossover point. In this case, another current point is found between the first two voltages. The third voltage used to find the current point is the initial voltage, V'_{i+1} , plus the minimum of the voltage change calculated during the previous timestep (the keyword DVTEST, Section 6.43.30, is used during the first iteration) and half of the difference between the limited voltage, found during the first step, and the initial voltage. A parabola is fitted through the three points and used to interpolate an estimated equilibrium potential where the total current will be defined to be zero. This method successfully dampens oscillating voltages and forces the problem to converge to a steady state solution. If the limited voltage is actually equal to V'_{i+1} , a small perturbation factor (VWIGGL, Section 6.43.30) is added to prevent numerical problems when the second stage derivative is calculated.

The set of surface voltages at the conclusion of this limiting process, V'_{i+1} , is used to calculate the second stage current derivative. The second stage derivative actually used to solve the second stage charging equation is defined to be the minimum (most negative) of the second stage derivative found using Eq. (4.56-3) and the derivative used during the first stage. Or

$$\left. \frac{dI}{dV} \right|_i' = \min \left[\left. \frac{dI}{dV} \right|_i, \frac{I'_{i+1} - I_i}{V'_{i+1} - V_i} \right]$$

The second stage may now be solved using ICCG to find V_{i+1} , the new surface voltage list.

4.56.30 PARTICLE BEAM CHARGING EFFECTS

Particle beam effects are accounted for in the surface charging model. Presently, particle beams are only seen as a current source (or drain) on the conductors. The interaction of returning beams with the object surfaces are not taken into account.

The beam's current contribution is added to the conductor to which it is attached until the conductor voltage is greater than the beam energy. The dI/dV term for the particle beam is discussed in detail in Section 4.51.

4.57 CHARGING MATRIX FORMULATIONS

To discuss the solution to Eq. (4.50-1), the notation of Section 4.56.10 is used to write

$$\underline{\underline{M}} \cdot \Delta \underline{V} = \underline{R} \quad (4.57-1)$$

where $\underline{\underline{M}}$ includes the capacitance, conductance and current derivative matrices. When the number of vector components is not too great (<1000), the Incomplete Cholesky Conjugate Gradient (ICCG) method is found to be an efficient means of solving Eq. (4.57-1) (see Section 4.30).

Experience has shown that ICCG is most effective when $\underline{\underline{M}}$ is diagonal or nearly diagonal (diagonal elements \gg off diagonal elements). To see how $\underline{\underline{M}}$ may be improved, write (4.57-1) as

$$(\underline{\underline{M}}_1, \underline{\underline{M}}_2, \dots, \underline{\underline{M}}_C) \Delta \underline{V} = \underline{R}$$

If surface 1 connects to infinity (subscript G) and conductor C, and considering only the capacitance,

$$\underline{\underline{M}}_1 = \begin{bmatrix} -C_{1G} & -C_{1C} \\ - & \\ - & \\ C_{1C} \end{bmatrix} \text{ and } \underline{\underline{M}}_C = \begin{bmatrix} C_{1C} \\ - \\ - \\ -C_{CG} \end{bmatrix}$$

where $C_{1C} \gg C_{1G}, C_{CG}$. (That this is a legitimate example, see the complete sample problem worked in 4.57.10.) It is the off diagonal entries of C_{1C} that need to be removed. This is accomplished by transforming (4.57-2) to

$$(M_1, M_2, \dots, M_C + M_1) \begin{bmatrix} \Delta V_1 & -\Delta V_C \\ & - \\ & - \\ \Delta V_C & \end{bmatrix} = R$$

A column transformation for each surface will "clean up" the upper right triangle of \underline{M} , while producing a transformed potential vector where surface potentials have been replaced by the potential difference between the surface and the underlying conductor. The lower left triangle of \underline{M} is diagonalized and symmetrized to the upper by similar row additions with corresponding transforms of \underline{R} on the right hand side. Currents (in \underline{R}) to surfaces remain unchanged, but currents to conductors are replaced as

$$I_C \rightarrow I_C - \sum_S I_S$$

where the sum is over all surfaces connected by capacitance or conductance to the main conductor, C and over the surfaces connected to their own underlying conductors; other conductors are treated as surfaces and referenced to C. The \underline{g} and \underline{V} on the right hand side are treated identically to \underline{g} , \underline{C} and \underline{V} on the left. Finally, the current derivative has terms in the same elements as the capacitance to infinity terms and is manipulated the same way.

Another feature of POLAR's charging model will change the previous transformations. This will occur when the routine computing the surface currents during the first stage (MAKJ1) finds the total current's sign is determined by the size of the electron secondary currents. In this circumstance the surface potential is controlled by the normal electric field. This may occur when a secondary or photoelectron space charge density becomes large enough to form a small barrier to the low energy portion of the emitted spectrum. Such a surface will be flagged and get its potential floated in the Poisson

solution according to the $d/dt (\vec{E} \cdot \hat{n}) = 0$ condition, while being held fixed during the circuit solution.

This fixing is not compatible with the \underline{V} transformation, so POLAR is forced to solve the circuit model with the original equations. Accuracy does not suffer, but ICCG will require more time for the same level of convergence.

Other conditions may also arise that will require a surface cell to be held at a fixed potential.

4.57.10 AN EXAMPLE OF MATRIX FORMULATION

Figure 4.57/1 shows the circuit diagram of an object which can be used to demonstrate the various matrix formulations. This object could be a 2 x 1 flat plate with two insulating surfaces (nodes 1 and 2) and two exposed conductors (nodes 3 and 4). By convention, the ground conductor is defined to be first named conductor, or in this problem, node 3. In the following example, the current derivative terms are left out. Since they would have appeared everywhere there was a capacitance to infinity term, they need not be carried through the example.

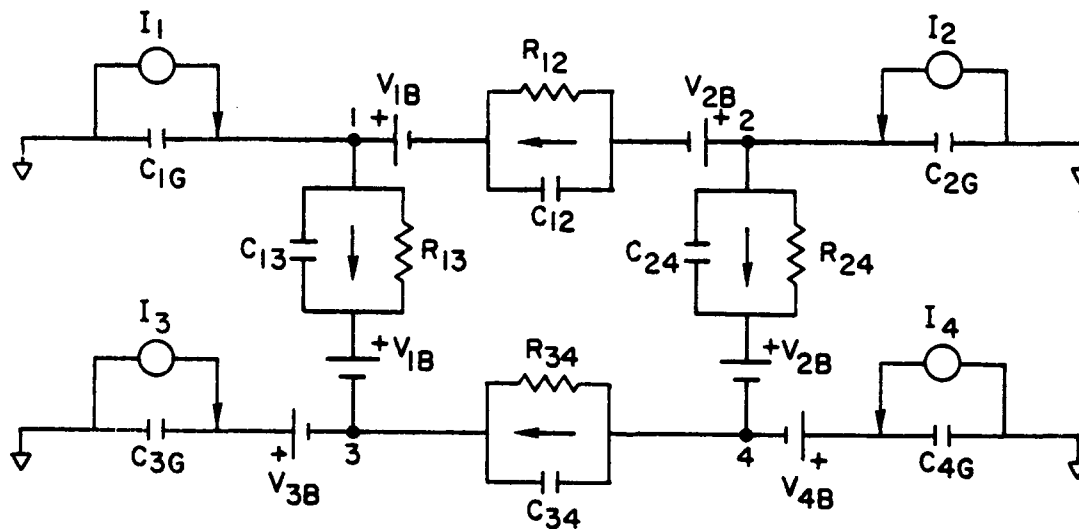


Figure 4.57/1. General circuit model used to study the matrix construction (see Figure 4.55/1 for a legend of circuit elements).

Using Kirchhoff's rules, the equations describing the current balance at each of the nodes can be written as

$$I_1 = -C_{1G} \frac{d}{dt} (0 - V_1) + (\sigma_{13} + C_{13} \frac{d}{dt}) [V_1 - (V_3 + V_{1B})] \\ - (\sigma_{12} + C_{12} \frac{d}{dt}) [(V_2 - V_{2B}) - (V_1 - V_{1B})]$$

$$I_2 = -C_{2G} \frac{d}{dt} (0 - V_2) + (\sigma_{24} + C_{24} \frac{d}{dt}) [V_2 - (V_4 + V_{2B})] \\ + (\sigma_{12} + C_{12} \frac{d}{dt}) [(V_2 - V_{2B}) - (V_1 - V_{1B})]$$

$$I_3 = -C_{3G} \frac{d}{dt} [0 - (V_3 + V_{3B})] - (\sigma_{13} + C_{13} \frac{d}{dt}) [V_1 - (V_3 + V_{1B})] \\ - (\sigma_{34} + C_{34} \frac{d}{dt}) (V_4 - V_3)$$

$$I_4 = -C_{4G} \frac{d}{dt} [0 - (V_4 + V_{4B})] - (\sigma_{24} + C_{24} \frac{d}{dt}) [V_2 - (V_4 + V_{2B})] \\ + (\sigma_{34} + C_{34} \frac{d}{dt}) (V_4 - V_3)$$

where

$$\sigma = \frac{1}{R}$$

If the indicated time derivative is performed, the magnetic field voltages across capacitors disappear (the various V_B terms). Writing the above equation in matrix form and using V_i in place of dV_i/dt ,

$$\begin{aligned}
 \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \end{bmatrix} &= \begin{bmatrix} C_{1G}+C_{12}+C_{13} & -C_{12} & -C_{13} & 0 \\ -C_{12} & C_{2G}+C_{12}+C_{24} & 0 & -C_{24} \\ -C_{13} & 0 & C_{3G}+C_{13}+C_{34} & -C_{34} \\ 0 & -C_{24} & -C_{34} & C_{4G}+C_{24}+C_{34} \end{bmatrix} \begin{bmatrix} \dot{V}_1 \\ \dot{V}_2 \\ \dot{V}_3 \\ \dot{V}_4 \end{bmatrix} \\
 &+ \begin{bmatrix} \sigma_{12}+\sigma_{13} & -\sigma_{12} & -\sigma_{13} & 0 \\ -\sigma_{12} & \sigma_{12}+\sigma_{24} & 0 & -\sigma_{24} \\ -\sigma_{13} & 0 & \sigma_{13}+\sigma_{34} & -\sigma_{34} \\ 0 & -\sigma_{24} & -\sigma_{34} & \sigma_{24}+\sigma_{34} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} \\
 &+ \begin{bmatrix} -\sigma_{12}-\sigma_{13} & \sigma_{12} & 0 & 0 \\ \sigma_{12} & -\sigma_{12}-\sigma_{24} & 0 & 0 \\ \sigma_{13} & 0 & 0 & 0 \\ 0 & \sigma_{24} & 0 & 0 \end{bmatrix} \begin{bmatrix} V_{1B} \\ V_{2B} \\ V_{3B} \\ V_{4B} \end{bmatrix}
 \end{aligned}$$

This is the fixed form of the matrices. When any node other than the ground conductor is fixed, the above matrix formulation is used. Note that the conductor magnetic field terms have dropped out of the matrices, thus the magnetic contribution vector can be changed.

$$\begin{bmatrix} V_{1B} \\ V_{2B} \\ V_{3B} \\ V_{4B} \end{bmatrix} \rightarrow \begin{bmatrix} V_{1B} \\ V_{2B} \\ 0 \\ 0 \end{bmatrix}$$

in order to simplify its appearance. The currents to the conductors, though, still depend upon surface voltage which includes the magnetic field voltage contribution.

As previously discussed in Section 4.57, it is desirable to diagonalize the matrix by moving the larger bulk capacities to the diagonal. To diagonalize the matrices, the rows and columns of the insulators are added to their underlying conductors. Then the conductors need to be added to the spacecraft ground conductor. As an example of the linear algebra involved, a simple case is worked out. A two variable linear equation is defined as

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

To add columns of a matrix without changing the equations,

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{11} + a_{12} \\ a_{21} & a_{21} + a_{22} \end{pmatrix} \begin{pmatrix} b_1 - b_2 \\ b_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

or adding rows,

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{11}+a_{21} & a_{12}+a_{22} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_1+c_2 \end{pmatrix}$$

and combining the results,

$$\begin{pmatrix} a_{11} & a_{11}+a_{12} \\ a_{11}+a_{21} & a_{11}+a_{12}+a_{21}+a_{22} \end{pmatrix} \begin{pmatrix} b_1-b_2 \\ b_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_1+c_2 \end{pmatrix}$$

Applying the above result to the fixed matrix form, the diagonalized or unfixed form can be written as

$$\begin{aligned}
 \begin{pmatrix} I_1 \\ I_2 \\ I_1+I_2+I_3+I_4 \\ I_2+I_4 \end{pmatrix} &= \begin{pmatrix} C_{1G}+C_{12}+C_{13} & -C_{12} & C_{1G} & -C_{12} \\ -C_{12} & C_{2G}+C_{12}+C_{24} & C_{2G} & C_{2G}+C_{12} \\ C_{1G} & C_{2G} & C_{1G}+C_{2G}+C_{3G}+C_{4G} & C_{2G}+C_{4G} \\ -C_{12} & C_{2G}+C_{12} & C_{2G}+C_{4G} & C_{2G}+C_{4G}+C_{12}+C_{34} \end{pmatrix} \begin{pmatrix} \dot{V}_1-\dot{V}_3 \\ \dot{V}_2-\dot{V}_4 \\ \dot{V}_3 \\ \dot{V}_4-\dot{V}_3 \end{pmatrix} \\
 &+ \begin{pmatrix} \sigma_{12}+\sigma_{13} & -\sigma_{12} & 0 & -\sigma_{12} \\ -\sigma_{12} & \sigma_{12}+\sigma_{24} & 0 & \sigma_{12} \\ 0 & 0 & 0 & 0 \\ -\sigma_{12} & \sigma_{12} & 0 & \sigma_{12}+\sigma_{34} \end{pmatrix} \begin{pmatrix} V_1-V_3 \\ V_2-V_4 \\ V_3 \\ V_4-V_3 \end{pmatrix} \\
 &+ \begin{pmatrix} -\sigma_{12}-\sigma_{13} & \sigma_{12} & -\sigma_{13} & \sigma_{12} \\ \sigma_{12} & -\sigma_{12}-\sigma_{24} & -\sigma_{24} & -\sigma_{12}-\sigma_{24} \\ 0 & 0 & -\sigma_{12}-\sigma_{24} & -\sigma_{24} \\ \sigma_{12} & -\sigma_{12} & -\sigma_{24} & -\sigma_{12}-\sigma_{24} \end{pmatrix} \begin{pmatrix} V_{1B}-V_{3B} \\ V_{2B}-V_{4B} \\ V_{3B} \\ V_{4B}-V_{3B} \end{pmatrix}
 \end{aligned}$$

Again, if the magnetic portion is multiplied out, the V_{3B} and V_{4B} cancel or

$$\begin{pmatrix} V_{1B} - V_{3B} \\ V_{2B} - V_{4B} \\ V_{3B} \\ V_{4B} - V_{3B} \end{pmatrix} \rightarrow \begin{pmatrix} V_{1B} \\ V_{2B} \\ 0 \\ 0 \end{pmatrix}$$

In other words, the magnetic field contribution to the circuit model is constant in both the fixed and unfixed forms.

Two more matrix formulations are possible with the inclusion of the biasing of two conductors to one another. They are the biased/fixed and biased/unfixed forms. Biasing reduces the four

variable problems to three by adding the equation $V_4 = V_3 + V_{bias}$. So if the substitution is made and the two conductor equations are added, the set of equations becomes

$$\begin{aligned}
 I_1 &= -C_{1G} \frac{d}{dt} (0 - V_1) + (\sigma_{13} + C_{13} \frac{d}{dt}) [V_1 - (V_3 + V_{1B})] \\
 &\quad - (\sigma_{12} + C_{12} \frac{d}{dt}) [(V_2 - V_{2B}) - (V_1 - V_{1B})] \\
 I_2 &= -C_{2G} \frac{d}{dt} (0 - V_2) + (\sigma_{24} + C_{24} \frac{d}{dt}) [V_2 - (V_3 + V_{bias} + V_{2B})] \\
 &\quad + (\sigma_{12} + C_{12} \frac{d}{dt}) [(V_2 - V_{2B}) - (V_1 - V_{1B})] \\
 I_3 + I_4 &= -C_{3G} \frac{d}{dt} [0 - (V_3 + V_{3B})] - (\sigma_{13} + C_{13} \frac{d}{dt}) [V_1 - (V_3 + V_{1B})] \\
 &\quad + C_{4G} \frac{d}{dt} (V_3 + V_{bias} + V_{4B}) \\
 &\quad - (\sigma_{24} + C_{24} \frac{d}{dt}) [V_2 - (V_3 + V_{bias} + V_{2B})]
 \end{aligned}$$

Note that the σ_{34} and C_{34} terms are lost upon addition. Or in matrix form, these become

$$\begin{aligned}
 \begin{pmatrix} I_1 \\ I_2 \\ I_3 + I_4 \end{pmatrix} &= \begin{pmatrix} C_{1G} + C_{12} + C_{13} & -C_{12} & -C_{13} \\ -C_{12} & C_{2G} + C_{12} + C_{24} & -C_{24} \\ -C_{13} & -C_{24} & C_{1G} + C_{13} + C_{4G} + C_{24} \end{pmatrix} \begin{pmatrix} \dot{V}_1 \\ \dot{V}_2 \\ \dot{V}_3 \end{pmatrix} \\
 &+ \begin{pmatrix} \sigma_{12} + \sigma_{13} & -\sigma_{12} & -\sigma_{13} \\ -\sigma_{12} & +\sigma_{24} & -\sigma_{24} \\ -\sigma_{13} & -\sigma_{24} & \sigma_{13} + \sigma_{24} \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix} \\
 &+ \begin{pmatrix} -\sigma_{12} - \sigma_{13} & \sigma_{12} & 0 \\ \sigma_{12} & -\sigma_{12} - \sigma_{24} & 0 \\ \sigma_{13} & \sigma_{24} & 0 \end{pmatrix} \begin{pmatrix} V_{1B} \\ V_{2B} \\ 0 \end{pmatrix} \\
 &+ \begin{pmatrix} 0 & 0 & 0 \\ 0 & -\sigma_{24} & 0 \\ 0 & \sigma_{24} & 0 \end{pmatrix} \begin{pmatrix} 0 \\ V_{bias} \\ 0 \end{pmatrix}
 \end{aligned}$$

where the magnetic field has disappeared from the matrix again. This is the biased/fixed formulation of the circuit equations.

To find the biased/unfixed formulation, the linear algebra techniques are applied as before to find

$$\begin{aligned}
 \begin{pmatrix} I_1 \\ I_2 \\ I_1+I_2+I_3+I_4 \end{pmatrix} &= \begin{pmatrix} C_{1G}+C_{12}+C_{13} & -C_{12} & C_{1G} \\ -C_{12} & C_{2G}+C_{12}+C_{24} & C_{2G} \\ C_{1G} & C_{2G} & C_{1G}+C_{2G}+C_{3G}+C_{4G} \end{pmatrix} \begin{pmatrix} \dot{V}_1-\dot{V}_3 \\ \dot{V}_2-\dot{V}_3 \\ \dot{V}_3 \end{pmatrix} \\
 &+ \begin{pmatrix} \sigma_{12}+\sigma_{13} & -\sigma_{12} & 0 \\ -\sigma_{12} & \sigma_{12}+\sigma_{24} & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} V_1-V_3 \\ V_2-V_3 \\ V_3 \end{pmatrix} \\
 &+ \begin{pmatrix} -\sigma_{12}-\sigma_{13} & \sigma_{12} & -\sigma_{13} \\ \sigma_{12} & -\sigma_{12}-\sigma_{24} & -\sigma_{24} \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} V_{1B} \\ V_{2B} \\ 0 \end{pmatrix} \\
 &+ \begin{pmatrix} 0 & 0 & 0 \\ 0 & -\sigma_{24} & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ V_{bias} \\ 0 \end{pmatrix}
 \end{aligned}$$

It should be pointed out that when these forms are used by the code, the magnetic field and bias (if appropriate) contributions are computed to become vectors then added to the current vector and the sum is manipulated into the appropriate form.

REFERENCES

- 4-1 Mandell, M. J. and I. Katz, "NASCAP Programmers' Reference Manual," S-CUBED Report SSS-R-84-6638, March 1984.
- 4-2 Parker, L. W. and E. C. Sullivan, NASA Report No. TN-D-7409, 1974.
- 4-3 Parker, L. W., "Calculation of Sheath and Wake Structure About a Pillbox-Shaped Spacecraft in a Flowing Plasma," Proceedings of the Spacecraft Charging Technology Conference, AFGL-TR-77-0051, NASA TMX-73567, 1977, ADA045459.
- 4-4 Laframboise, J. G. and L. W. Parker, "Probe Design for Orbit-Limited Current Collection," Phys. of Fluids, 16, N5, 1973.
- 4-5 Kershaw, D., "The Incomplete Cholesky-Conjugate Gradient Method for the Iterative Solution of Systems of Linear Equations," J. Comp. Phys., 26, p. 43, 1978.
- 4-6 Katz, I., D. E. Parks, M. J. Mandell, J. M. Harvey, D. H. Brownell, Jr., S. S. Wang and M. Rotenberg, "A Three Dimensional Dynamic Study of Electrostatic Charging in Materials," NASA CR-135256, August 1977.
- 4-7 Hackenberg, O. and W. Bauer, Advances in Electronics and Electron Physics, 16, p. 145, 1962.
- 4-8 Feldman, C., Physical Review, 117, p. 455, 1960.
- 4-9 Ashley, J. C., C. J. Tung, V. E. Anderson and R. H. Ritchie, (i) AFCRL-TR-75-0583, ADA019507; (ii) RADC-TR-76-220, ADA029449; (iii) RADC-TR-76-125, ADA025488; (iv) IEEE Transactions on Nuclear Science, NS-25(6), p. 1566, 1978.
- 4-10 Darlington, E. H. and V. E. Cosslett, "Backscattering of 0.5-10 keV Electrons from Solid Targets," J. Phys., D5, p. 1969, 1972.
- 4-11 Shimizu, R., "Secondary Electron Yield with Primary Electron Beam of Kilo-electron-volts," J. Appl. Phys., 45, p. 2107, 1974.
- 4-12 Handbook of Mathematical Functions, U.S. Department of Commerce, National Bureau of Standards, Applied Math Series, 55, p. 924, 1964.

REFERENCES

(continued)

- 4-13 Cousinie, P., N. Colombie, C. Fert and R. Simon, "Variation du coefficient d'émission électronique secondaire de quelques métaux avec l'énergie des ions incidents," Comptes Rendus 249, p. 387, 1959.
- 4-14 Hill, A. G., W.W. Buechner, J.S. Clark and J.B. Fisk, "The Emission of Secondary Electrons Under High-Energy Positive Ion Bombardment," Phys. Rev., 55, p. 463, 1939.
- 4-15 Foti, G., R. Potenza and A. Triglia, "Secondary-Electron Emission from Various Materials Bombarded with Protons at $E_p < 2.5$ MeV," Lett. al Nuovo Cim., 11, p. 659, 1974.
- 4-16 Aarset, B., R.W. Cloud and J.G. Trump, "Electron Emission from Metals Under High-Energy Hydrogen Ion Bombardment," J. Appl. Physics., 25 p. 1365, 1954.

5. POLAR CODE STRUCTURE

This chapter is designed to provide insight into the internal workings of POLAR. Whenever possible, actual subroutine names and variable names will be used.

5.10 TOP DOWN VIEW OF THE POLAR PACKAGE

POLAR is actually four standalone programs, and several utility programs, that communicate through a minimum number of files. This approach allows a high degree of flexibility in model building while minimizing the amount of unnecessary computing. These programs are VEHICL, ORIENT, NTERAK, and SHONTL. Their functions are described below. Whenever scratch files are used, they are assigned and disposed of automatically. In general, only two files are needed to allow communication between the four modules.

VEHICL is the object definition program. It utilizes much of the user-oriented object definition procedures developed at S-CUBED for NASCAP. With VEHICL, one uses a variety of basic building blocks to define the vehicle to be modeled on a variable sized 3-D grid. One also defines all of the surface properties and underlying conductors. VEHICL then completes the vehicle electrical model and creates a number of "connectivity" tables to accelerate NTERAK execution. This information is written on two files, 11. and 19., which carry this information to the other modules.

ORIENT is the attitude control program. User input is simply the dominant plasma flow direction viewed from the vehicle. If necessary, ORIENT will rotate the vehicle and object grid so as to keep the wake direction predominantly in the +Z direction. ORIENT will also restructure most of file 11 in order to organize the slices properly. A set of six ORIENT runs could catalog all of the possible coordinate orientations, and allow for all Mach vectors.

NTERAK is the biggie that actually calculates the vehicle-plasma interaction. The internal workings and I/O are the subject of most of this document, but it should be emphasized that once a Mach vector has been defined, the extended computational grid will be "burned" into file 11. The file retains a complete restart-continue capability, but a fresh file should be used if the Mach vector or the grid dimensions are to be changed.

SHONTL is the machine independent plotting package. It is designed to be run semi-interactively. By this we mean that plotting directives are entered by a "keyword" input system that functions in both batch and interactive environments, but nothing is plotted until the session is concluded and the appropriate plotting utility is executed. This somewhat cumbersome procedure is necessary to maintain the machine independence of SHONTL. SHONTL can be used after any of the previous three modules to graphically check the work progress at any given stage. SHONTL reads from file 11, and communicates with the device drivers through file 2. The file 2 contains the generic pen-move and vector commands written on file 2 by SHONTL which are later translated to a specific graphics display device.

SHONTL is also a powerful debugging and diagnostic tool. It can be used to produce hidden line plots, print data stored using the buffered I/O routines and can be used to see up diagnostic or test runs.

Two utilities are provided for pre- and post-processing POLAR calculation, SUCHGR and TRMTLK. Several device dependent graphics programs which display the contents of file 2 are included. There are also support tools available to handle job control tasks and software maintenance of the source code.

5.11 VEHICL

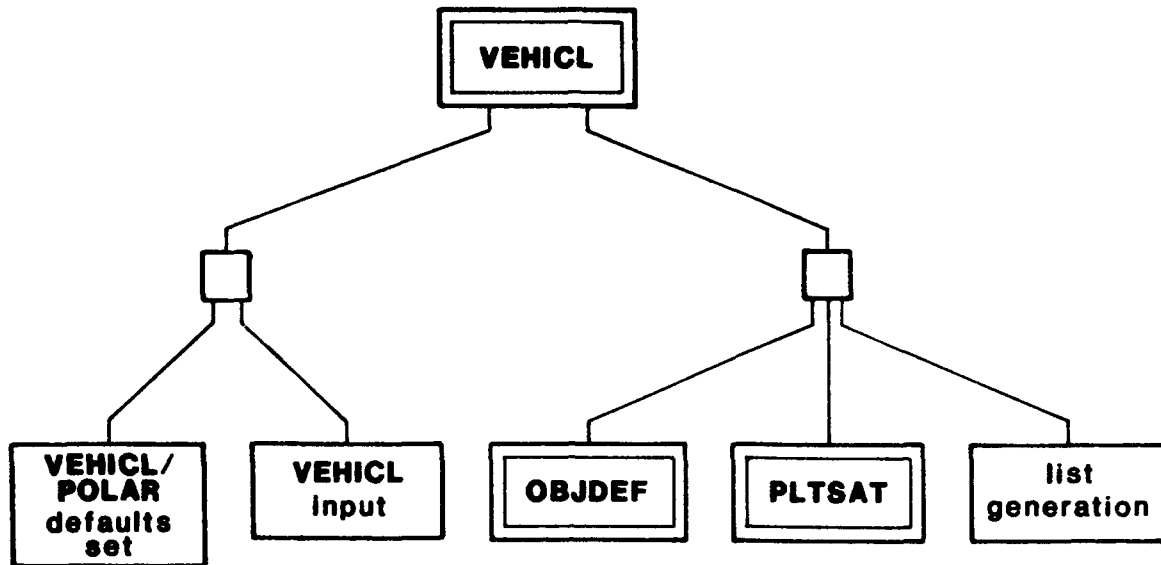


Figure 5.11/1. VEHICL structure.

The vehicle or object definition phase of POLAR is done once for each object by the use of VEHICL. Figure 5.11/1 depicts the organization of VEHICL. The subroutine tree is traversed from left to right.

The first step is to set local VEHICL default parameters. The default values to be used later during NTERAK are set by VEHICL. This is also done during the default setting phase.

After setting defaults, the routine VEHDEF solicits input from standard input (unit 5 in standard Fortran) in either a batch or interactive mode.

When the end of the keyword input portion has been completed, the OBJDEF subroutine is called to read the object description from file 20 and initialize the element and surface lists.

Upon successful completion of satellite definition, PLTSAT generates plots of the satellite if desired. Hidden line perspective plots and/or axial material plots can be generated.

During the initial object definition, general tables and lists are created. The final phase of VEHICL is preprocessing these lists to generate the various tables used by NTERAK to speed up the calculations. The generation of these detailed lists imposes a strict set of object validity tests and frequently errors in object definitions are found in this part of VEHICL.

5.12 ORIENT

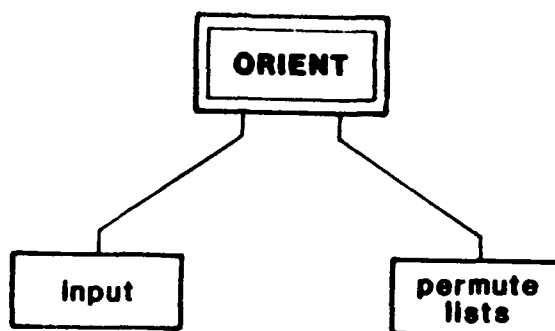


Figure 5.12/1. ORIENT structure.

Because the modeling of plasma flow is facilitated by the use of a preferred direction during computations, the plasma flow is defined to be in the +Z direction in NTERAK. This is not always the most convenient way to define an object. ORIENT provides the means to rotate an object to a preferred orientation from the arbitrary orientation used during definition.

The first step is to solicit input from standard input in either a batch or interactive mode. After command input is complete, the lists generated by VEHICL are permuted to the new set of axis.

ORIENT can be called several times in sequence to perform many permutations. The final lists are in the same form as those generated by VEHICL. The ORIENT can, of course, be skipped altogether if the object is already properly oriented or no flow is going to be used.

5.13 NTERAK

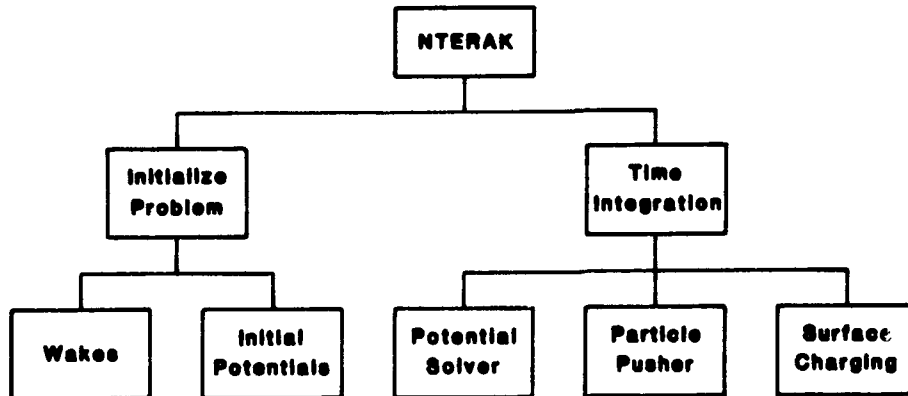


Figure 5.13/1. Structure of NTERAK module.

NTERAK has two major divisions, the initialization processes and the time integration sections. Figure 5.13/1 depicts these portions of NTERAK.

The first calculations needed are those which define the wake structure and the initial surface and space potentials. The details of the problem determine the complexity, or even the necessity of these computations.

The majority of interest in most cases is upon the results of the time integration portion of NTERAK. NTERAK iterates between the potential solver, particle pusher, and surface charger to find a quasistatic solution for a problem.

The potential solver, PWASON, uses the space charge densities and the surface potentials to compute space potentials. The particle pusher, CURREN, uses the space potentials to calculate a particle sheath and to compute electric fields to push the attracted specie of particles from the sheath to the object surface. The particle pusher computes new space charge densities within the sheath and new particle currents to the object's surfaces. The surface charger, CHARGE, uses the particle currents to compute new surface potentials. The new surface potentials and space charge densities are then used again by the potential solver to continue the iteration.

Each of the three modules are independent of one another and if one of them is not necessary for a calculation, it can be left out of the iteration cycle.

NTERAK can be stopped at any point during initialization or the time integration using the keyword input. Input is accepted in either a batch or interactive mode from the standard input file. The input is used differently than in VEHICL or ORIENT. Keywords are read until one of the calculations pictured in Figure 5.13/1 is requested. At that point input is suspended and the computation is performed. At the completion of the calculation, input is resumed. In this manner, input parameters can be changed from iteration to iteration.

After any calculation, NTERAK can be stopped and the intermediate results viewed using SHONTL or whatever. The run can be continued by starting NTERAK again with the appropriate instructions.

5.14 SHONTL

The SHONTL module is described in detail in Section 5.82. The SHONTL module is used to produce plots, print grid and list data stored on the buffered I/O files 11 and 19, make hidden line perspective pictures of the object, and serve as a diagnostic, debugging, and general all purpose driver.

5.15 UTILITIES

In addition to the four main POLAR executables, there are several specialized utilities to aid the analysis of spacecraft/environment interactions. The utilities provide the means to perform quick calculations before beginning NTERAK jobs, the ability to study the charging of the individual spacecraft surfaces, interfaces to two different standard graphics packages, a method of NTERAK batch job control, and some tools to maintain and modify the source code for the POLAR package.

SUCHGR performs quick analysis of the interaction between materials and POLAR environments. This utility contains both orbit limited and space charge limited algorithms for calculating the equilibrium surface potential, surface currents and sheath radii. Typically SUCHGR calculations are used to anticipate the charging behavior of surfaces, ignoring geometric and surface/surface interaction effects, before beginning NTERAK computations.

After NTERAK charging calculations, TRMTLK can be used to provide information on the charging history of individual surfaces and conductors. The surface voltage charging history, current breakdowns for each insulating surface, and other items are taken from file 16 and displayed in both table and plot formats. Tools are available to select groups of surfaces or individual surfaces with out requiring surface numbers.

Two types of standard display device plots are supported by this version of POLAR. Plot files are translated to Tektronix 4014 graphics commands by T4014. Postscript versions of file 2 can be created by PSTPLT. For computing environments where the calculations are performed on one machine and graphics are viewed on another incompatible machine (for example NTERAK is run on a batch oriented CRAY and plots are viewed on a VAX system), a pair of routines can be used to transfer plotting data in an ASCII format. READ02 converts the contents of file 2 (an unformatted file) to a text file, file 4. The text file can then be moved to a second machine and displayed in Tektronix 4014 graphics commands with PLOT04.

Since NTERAK calculations can take a fair amount of time, sometimes it is desirable to stop a running batch gracefully and then restart it again later. At convenient intervals, NTERAK checks for job control input files (.JCI files). STOPRUN sets a flag to indicate the desire for a graceful termination of the current run. KILLJOB sets a flag to kill the run immediately upon detection.

The source code for POLAR is normally organized in a directory tree, where each library and executable have their own subdirectories of source code. A shell script, MAKEP1, is used to compile and load the POLAR executables. A specially tailored version of MAKEP1 is available for making modifications of POLAR without incorporating them into the main version of the POLAR package.

5.20 SLICE GRID SYSTEM

The displaced slice grid system is designed to provide the computational space in which to solve Poisson's equation for the shuttle orbiter, including a wake extending many spacecraft lengths. Hence the grid must continue for an arbitrary length in the plasma flow direction.

To facilitate this, the grid is composed of a variable number of XY slices, stacked along the Z axis, rather like a loaf of sliced bread. Figure 5.20/1 illustrates a displaced grid system along with a number of important parameters. Objects are defined on a rectangular NXOB x NYOB x NZOB grid (5.11, 6.20). In all cases, the object grid will have been rotated by ORIENT such that the dominant component of the plasma flow vector VMACH is oriented along the +Z direction. As shown in the figure, the grid follows VMACH by stepping ± 1 unit in the X and Y direction every IDELX and IDELY mesh units along the Z direction (the ± 1 step follows the sign of IDELX or IDELY). These step intervals are calculated in the routine SPACE according to the relation (FORTRAN)

$$\text{IDELX} = \text{VMACH}(Z) / \text{VMACH}(X) \ast 0.5$$

$$\text{IDELY} = \text{VMACH}(Z) / \text{VMACH}(Y) \ast 0.5$$

where the \ast follows the sign of VMACH(X). Since IDELX is an integer, the 0.5 centers the velocity ratio between integral increments.

The computational grid must enclose the object grid with a minimal amount of wasted space. To accomplish this, the routine SPACE references the two grids at the point shown in Figure 5.20/1, then calculates the NXGRTH(NYGRTH) necessary to fit the grids together. When the user anticipates the need for additional work space, the INPUT parameters NXADNT, NXADNB, NYADNT, and NYADNB can increase the

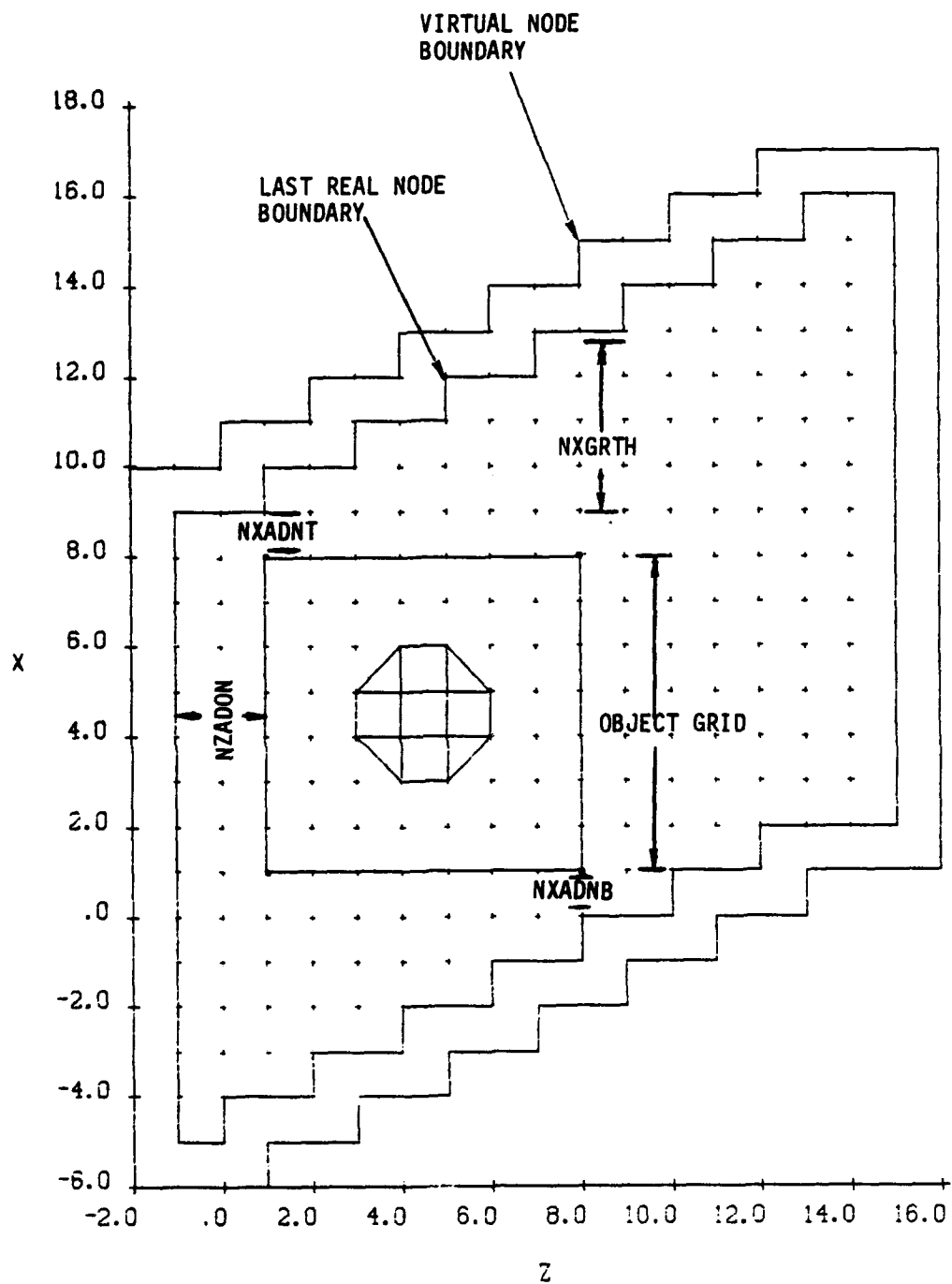


Figure 5.20/1. A X-Z cut of a typical NTERAK computational mesh showing the object definition grid and the computed (NXGRTH, IDELX) and user-specified (NXADNT, NXADNB, NZADON, NZTAIL) grid parameters.

X-Y grids without requiring a redefinition of the object space. The GRTH and ADON parameters are included into the final NX x NY dimensions.

The computational space is characterized by the following parameters:

NXOB,NYOB,NZOB	=	The real node dimension of the object grid along the X, Y, and Z directions.
NX	=	$NXOB + NXGRTH + 2 \times NXADON$
	=	the real node dimension of a slice along the X direction.
NY	=	$NYOB + NYGRTH + 2 \times NYADON$
NZ	=	$NZOB + NZTAIL + NZADON$

The NZADON and NZTAIL are also inputs to NTERAK and are currently limited to total a Z node count of 100. This limit could be extended indefinitely subject to available disk storage and budget limitations. This last feature is the intended result of NTERAK's data management system. This system allows models to be constructed primarily on disk with the computer "core" used to perform arithmetic on small volumes of space.

5.21 SLICE MACHINERY

The grid machinery consists of routines designed to move or page grid information between disk and core. In both media, information for each of the vectors ('p', 'r', 'u', etc. (4.21.1) are called vectors even though they are scalars at a particular grid point) involved in a calculation is organized into individual one-dimensional arrays corresponding to each X-Y slice.

NTERAK distinguishes between two types of nodes, real and virtual. At real nodes a problem's variables are truly variables. Virtual nodes exist as the outer boundary of the problem, where values are generated according to the boundary conditions. There may be one or two virtual nodes beyond the edge of a slice (see Figure 5.20/1) depending on the proximity of a step. Only real nodes are paged in and out. Each vector slice has a real node length $NX * NY$ with the assumed convention that the X coordinate varies the fastest.

In order to relate the slices to one another and perform arithmetic with minimal confusion, it is necessary to adopt a standard coordinate system. There are two possible choices; object and slice coordinates. We have chosen object coordinates as the primary system, but slice coordinates are sometimes required by lower level routines. Object coordinates reference the "least" real node of the object grid as (1,1,1) (see Figure 5.20/1). By "least" we mean the node for which $NX * NY * (Z-1) + NX * (Y-1) + X$ is a minimum. Thus the least real node object coordinates of some slices may be less than zero, while the slice coordinates of the least node will be (1,1,ZSLICE) where ZSLICE numbers from 1 to NZ. The relationship of the slices to each other is written in the common block MESHY by the routine GRID. This block contains the object coordinates of the least real node of each slice. For further detail concerning subroutines and common blocks see Appendices A and B.

NTERAK's grid machinery exchanges information between disk and a special common block called CBUF. The core location of CBUF is such that its addresses are the highest of all other data and instructions. On many machines this allows the CBUF length to be extended or reduced dynamically during execution to precisely match storage requirements. The allocation and addressing of data space in CBUF is controlled by three routines (see Section 5.30), MRBUF, BUFCLR, and BUFSET. MRBUF is called only once from the level 2 routine SPACE and initializes the array VPROPS which contains all of the individual vector properties that affect the vector's handling storage requirement in CBUF and on disk.

Subsequently, any computational process requiring vector slice transfers to/from disk will first call BUFCLR. BUFCLR clears the ADDRES common block which contains the CBUF addresses, releases any dynamical requested core, and resets other storage control variables.

The next step is to call the routine BUFSET, passing it a list of up to four vector names and the number of slices that will be needed in core. BUFSET will assign each vector a region in CBUF, and list this addressee in the common ADDRES. BUFSET may be called repeatedly for a total of 12 vectors. For example, in an operation requiring three slices of 'R', 'R' would be assigned the address sequence; A, B, C, A, B, C, A, B, ... , for all 1-NZ slices. The actual transfers are effected by PAGER, which can read, write, or read-write slices. Suppose that 2, 3 and 4 were in core starting at the CBUF address B, C and A, respectively. If PAGER were called to read-write 'R', for slices 3-5, 'R' slice 2 at B would be written on disk, and slice 5 would be read into CBUF at B. The R slices 3 and 4 would remain untouched.

Some other routines that are commonly used with this grid machinery should be mentioned. One is XYGRID, which looks in the MESHY common block and returns the X and Y limits of a slice in object coordinates. The others are XBNDRY, YBNDRY, and ZBNDRY. These routines will take an element index (equal to the least index of its eight nodes) and determine which of the element's eight nodes are real or virtual (boundary nodes). This information is kept in the array MBXYZ(8). To form an element on the staggering grid, VERTIO is called to pick four nodes from each of the two bordering slices totaling eight vertices. Individual words of a vector at a node are located in CBUF by the statement function (or integer function on some machines) IPUFAD.

In retrieving these words, the MBXYZ array is consulted to determine if a node is virtual. If so, the boundary value (currently zero) will be used for that node.

The routine GETSCR is a cousin to VERTIO. It will 'GET' or 'REP' the element centered quantities 'GI' or 'SCRN'. 'GI' is the name given to the normalized ion densities (3.20, 4.41), and 'SCRN' refers to screening factors (4.43).

5.22 VOLUME ELEMENT MACHINERY

To illustrate NTERAK's volume element machinery we will describe its use by the subroutine COPROD. COPROD's function is to generate the $\underline{M} \cdot \underline{U}$ product and $\underline{U} \cdot \underline{M} \cdot \underline{U}$ inner product (4.30). Once slice information has been accessed and resides in core (5.21), COPROD must calculate residuals, element by element, and return the vector so calculated back to the disk in slices. These operations are complex and require fairly elaborate machinery. We begin by offering a brief overview.

COPROD begins by reading in the relevant slice information, establishing a section of the computational space in core. This volume is swept, element by element. Each element is characterized by the coordinates of its lowest indexed vertex. The potentials, and other vector information, for each of the eight vertices of a particular element are extracted from the main /CBUF/ array by VERTIO. VERTIO also replaces or augments array entries with calculated vertex information.

The potentials at the boundary of the computational space are assumed to be fixed and known (presently set to zero). Hence they are not stored explicitly in CBUF. Instead, COPROD examines each vertex of each volume element and determines if any be an implicit boundary. Those that do are fixed at the boundary potential. VERTIO (5.21, Appendix B) takes the X-Y displacement of the slices into account in picking out the vertices. Having set the potentials at the vertices of a particular element with VERTIO, COPROD calls ELEMNT to look up its characteristics in the list 'LTBL'. LTBL is an array with an entry for each element in the object grid, 'LTBL'. LTBL is an array with an entry for each element in the object grid, containing the following bit coded information (5.23): the element type, the number of surface cells sharing one or more nodes with the element (NCELLS), the element orientation, and the top/bottom flag.

The number of surface cells (NCELLS) sharing nodes with the element is used to refer to a second list, LCEL. If NCELLS is non-zero, DCVCEL is called to decode the next (NCELLS + 1) entries in the LCEL (5.25) list. The first word of this group will be an element I.D. number used merely as a check. The remaining words are also bit coded with: the surface cell number, the NODCOD telling which nodes are shared with the surface, and the FCN number (4.21) telling which element face, if any, the surface occupies.

If a vertex or node is shared by a surface cell, its potential is replaced with the surface potential for that cell, stored in the array SVRFV, sequentially by cell number. In the same way the contribution to the residual derived from the shared vertex is returned to the list SVRFR rather than the residual vector. Hence the surface potentials play the part of additional grid points, or variables in the matrix conjugate gradient equations. If a vertex is shared by more than one surface cell, the adjoining cell potentials are averaged by FORFIL (4.21) and assigned temporarily to the vertex. Once the vertice potentials have been collected, the residuals (4.30) are calculated by either PCUBES or ECUBES (4.20) and shared back to surface cells SRFR entry.

5.23 ELEMENT TABLE, LTBL

The element table, LTBL, is a list with one entry for every element in the grid. An element's LTBL entry can be accessed by calculating its relative address as if it were a 3-D array LTBL (X,Y,Z) where the Z range starts with the first slice in core (5.21).

Each word is coded in the following manner:

```

31 | 10 9 8 | 17 6 5 4 3 2 1 0 9 | 18 7 6 5 4 3 | 12 1 0 |
   D       C       B       A

```

where:

- A - The element-type number.
- B - The number of surface cells sharing nodes with the element (up to 15) (NCELLS).
- C - The orientation of the cell. This is only significant for partially filled cells and is explained below.
- D - Top/Bottom flag for elements above or below a thin plate.
 - D = 1 - bottom
 - = 2 - top
 - = 3 - both top and bottom, type 5

The following element types are allowed:

<u>Type Number</u>	<u>Bits</u>	<u>Description</u>
0	000	Empty cube
1	001	Half-empty wedge
2	010	Cube with diagonal on one face
3	011	Tetrahedron
4	100	Truncated cube
5	101	Empty cell bisected by a diagonal thin plate
6	110	Unused
7	111	Filled cube

These volume elements are described in 4.21.

The orientation code is a nine-bit (three octal digit) code describing how a non-symmetric element may be transformed into its "standard" orientation. The transformation (consisting of rotations, inversions, and translations) to the "standard" orientation is that transformation

which takes vector \underline{r} into vector \underline{s} , where

$$\underline{r} = (x, y, z)$$

$$\underline{s} = (q(i_1), q(i_2), q(i_3))$$

i_1 is the octal digit in bits 14-12

i_2 is the octal digit in bits 11-19

i_3 is the octal digit in bits 8-6

$$q(1) = x \qquad q(5) = 1-x$$

$$q(2) = y \qquad q(6) = 1-y$$

$$q(3) = z \qquad q(7) = 1-z$$

For example, the octal code 365 implies the transformation

$$\underline{s} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ -1 & 0 & 0 \end{pmatrix} \underline{r} + \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

5.24 SURFACE CELLS

5.24.1 SURFACE CELL LIST, KSURF

One of the lists produced by VEHICL and stored on file 11 (see Chapter 5.30 on file 11) is the surface cell list, KSURF. This list consists of bit packed word pairs for each surface cell. The information content is explained below, and the bit coding convention is illustrated in Figure 5.24/1. The bit ordering notation used here assigns each bit the exponent of 2 required to move an integer 1 to that location by multiplication.

COND	=	Conductor number, 1 to 15.
NORM	=	Surface normal in Miller indices, two bits for each index. The lowest bit representing 1 or 0, and the highest set for minus, off for plus.
XI,YI,ZI	=	Lowest coordinates of the volume element with which the surface cell is associated (the element that the cell normal points in to).
MAT	=	Material number, assigned by order of definition.
B	=	Set for all right-triangular surface cells (100,010, etc.) and for all equilateral (111) triangles whose enclosing volume element is mostly empty.
H	=	Orientation code for right-triangle surface cells. The two bits define the location of the right-angled corner in the plane of the triangle, i, j. The greatest bit refers to j. The j and i are related to the normal direction as follows:

<u>NORM</u>	<u>i</u>	<u>j</u>
Z	X	Y
X	Y	Z
Y	Z	X

X0,Y0,Z0	=	Volume element that the surface normal points out of.
TB	=	0 - not applicable
	=	1 - bottom surface of a thin plate
	=	2 - top

The KSURF list is written on file 11 by either subroutine VESURF or ORSURF. Prior to output it is 'Z' ordered to produce sliceability in the chosen orientation. It is written on file 11 in a series of records with each record containing all the word pairs with a given ZI coordinate. These records are indexed by the NWSURF(Z) list in record 1 of file 11. For each ZI of the object grid there is a NWSURF entry set equal to the number of KSURF words in the record for that ZI. If an NWSURF entry is zero, their corresponding null ZI record of KSURF is skipped.

<u>BIT</u>	<u>KSURF (1, N)</u>	<u>KSURF (2, N)</u>
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		
27		
28		
29		
30		
31		

COND

MAT

NORM

ZN

B

YN

XN

H

XI

X0

YI

Y0

ZI

Z0

TB

Figure 5.24/1. KSURF surface cell list bit code.

5.25 LCEL, CONNECTIVITY

The LCEL list is used in the potential solving segment to connect node point to surface cells (4.20, 5.22). This is a bit packed list with the following structure:

```

    ELT#
    CODED WORD
    CODED WORD
      .
      .
      .
    ELT#
  
```

The ELT# is a coded element identifier,

$$\text{ELT\#} = 4096 \times (I + 64 \times J + 4096 \times K)$$

which serves as a check for correct addressing in the list. Addressing is accomplished by accumulating the NCELLS word count from the element table LTBL (5.22, 5.23).

The coded words are bit packed as follows:

31 0 | 9 8 7 | 6 5 4 3 2 1 0 | 9 8 7 6 5 4 3 2 | 1 0 9 | 8 | 7 6 5 4 3 2 1 0 |

D

C

B

A

where:

- A - Bits 0-7 are set if the corresponding vertex is shared with the surface cell. The vertices are numbered with X changing faster than Y which changes faster than Z, e.g., for element 1, 1, 1

<u>Number</u>	<u>Coordinates</u>		
	<u>X</u>	<u>Y</u>	<u>Z</u>
1	1	1	1
2	2	1	1
3	1	2	1
4	2	2	1
5	1	1	2
6	2	1	2
7	1	2	2
8	2	2	2

- B - Is the standard orientation surface node number (see 4.21) that the surface will have when forming the element into which this surface points.

- C - Is the surface number.

- D - Is a code for surfaces that are on the top or bottom of a thin plate (4.21.1). The code is the same as the one used by the KSURF entries (5.24.1).

0 - not applicable

1 - bottom surface

2 - top surface

5.30 FILE SYSTEM

One of the most outstanding features of POLAR is its file management system. This system, combined with the sliced grid system (5.20) allows POLAR to model variably large 3-D problems (20,000 grid points is common), with fewer than 100,000 words of core. The modules VEHICL, ORIENT, NTERAK, and SHONTL communicate through the mass storage files 11 and 19 (see 5.31). Internal to each module, other files of differing types are used as scratch files. VEHICL uses the NASCAP object definition coding and thus a variety of NASCAP files are used initially in VEHICL as scratch files (Reference 1). Two files (11 and 19) are used instead of just one, so that one may be indexed with literal name keys (19), and the other (11), with number type index keys. Additional charging information is kept on file 16. This file can be interpreted after NTERAK runs using the TRMTLK utility.

5.31 MASS STORAGE FILE MANAGEMENT

Data stored on the permanent file 11 and 19, and the scratch file 9 and 10, is written and retrieved using the CDC MSIO system (mass-storage-input-output). On non-CDC machines, the MSIO routines OPENMS, WRITMS, READMS, and CLOSMS are provided by POLAR FORTRAN routines which mimic the CDC versions to provide a machine independent interface to the host system. We now describe these four routines as they are used in POLAR.

OPENMS:

Example: DIMENSION IND (NUM)

 CALL OPENMS (LUN, IND, NUM, t)

LUN - The logical unit number of the file.

IND - Space provided for a working copy of the file index.

NUM - Length (words) of IND

t - type of index

 = 0, number type index

 = 1, name type index

To index XR records, a $t = 0$ file requires $NUM \geq XR + 1$;

for $t = 1$, $NUM \geq 2 \cdot XR + 1$.

READMS:

Example: DIMENSION BUFR(1000)

 CALL READMS (LUN, BUFR(100), NWDS, KEY)

LUN - Logical unit number

BUFR([B0]) - Starting address to which data is to be read

NWDS - Number of words to be read

KEY - The index key under which the data record was stored.

WRITMS:

Example: `CALL WRITMS (LUN,BUFR,NWDS,KEY,r)`
LUN, NWDS, and KEY are as for READMS; data is read from BUFR. $r = -1$, always for POLAR. This specifies that a record may be overwritten only if the new record length does not exceed the old length. When this occurs, a new record is written and a link stored in the old location.

CLOSMS:

Example: `CALL CLOSMS (LUN)`
This routine writes the working index from core to the file copy. It is called frequently to maintain an up to date file copy of the index to protect data against unexpected crashes and stops.

5.32 MRBUF PLUS BUFSET PLUS FRIENDS

To enhance the efficiency of both the execution and development of POLAR, a formalized system of variable characterization and identification is used. This begins with the array VPROPS(20,70) wherein 20 different properties of 70 different variables are initialized by MRBUF. To set up a work space for a variable (or a slice of a variable) in the general purpose buffer CBUF, a call is made to BUFSET using the literal name of a variable (5.33). BUFSET will move the vector properties from VPROPS to a working array (IAVECS(20,I)) and the index I will be written into the MELAD common block in the variable pseudonym. For example

```
CALL BUFSET(5, 'POT', ...)
```

would result in space for five Z slices of potential to be allocated in CBUF with all essential information stored in IAVECS (20,IPOT) with IPOT being found in the MELAD common block.

The IAVECS properties are listed below, with a V indicating a non-variable property that is placed in VPROPS by MRBUF.

V	IAVECS(1,I)	=	Name
	IAVECS(2,I)	=	Start address in CBUF
	IAVECS(3,I)	=	Number of slices
V	IAVECS(4,I)	=	Vector length of a slice
V	IAVECS(5,I)	=	Words per vector

(The number of words in a slice is IAVECS(4,I) * IAVECS(5,I).)

V	IAVECS(6,I)	=	Mass storage file number
	IAVECS(7,I)	=	Low pointer (shifted slice coordinates), see property 14 and Section 5.21 and
	IAVECS(8,I)	=	High pointer used by the data paging routine PAGER
	IAVECS(9,I)	=	Lower (object coordinates, 5.21) and
V	IAVECS(10,I)	=	Upper object limits, inclusive
	IAVECS(11,I)	=	Starting key number for the number key files.
V	IAVECS(12,I)	=	X dimension of a slice
V	IAVECS(13,I)	=	Type of slice or vector. Choices are: 'VARI' - variable length (example, LCEL) 'SNGL' - single slice (example, SRFV) 'WORK' - work space 'OTHR' - other (e.g., FOTO) 1 - object slice (e.g., LTBL) 2 - real grid (e.g., POT) 3 - outer virtual node (e.g., LCOF) 4 - virtual element grid (e.g., DION)
V	IAVECS(14,I)	=	Shift added to slice coordinates to prevent FORTRAN MOD function of negative or zero slice coordinates
V	IAVECS(15,I)	=	Y dimension of slice

- V IAVECS(16,I) = Lower Z limit of the variable range
- IAVECS(17,I) = Pointer to relate node data to auxiliary node
 data (necessary for double points created by
 thin plates)
- V IAVECS(18,I) = Word type, 'REAL', 'INTE' (integer), 'OCTL',
 'ALPH'

5.33 MRBUF VARIABLE LIST

<u>NAME</u>	<u>DESCRIPTION (Section Reference)</u>
AMAT	A work space used for calculating RMAT
ASRF	Alt. of KSURF in CBUF, unsliced
AXDE	Auxiliary DELC values
AXDI	Auxiliary DION values
AXGH	Auxiliary GH values
AXGI	Auxiliary GI values
AXQU	Auxiliary QUSD values
AXRE	Auxiliary RHOE values
AXRI	Auxiliary RHOI values
AXSC	Auxiliary SCRN values
CLRG	Vector list of large capacitances of insulating surfaces
CSML	Vector list of small capacitances to insulating surfaces and conductors
DD	Unused
DELC	Electron density array
DION	Composite ion density array
EBAK	Electron backscattered current to each surface
EHOP	Electron hopping current for each surface
ENRM	Normal electric field to surface
ESEC	Electron secondary current to each surface
EXTV	Surface voltages found during first trial step
FOSH	Surface shadowing information
FOTO	Memory of past charge information (not implemented yet)
GH	Neutral h^+ density (with respect to its own species)

GI	Neutral ion densities
ICND	Conductor number of each surface
IFIX	List indicating surfaces with fixed voltages in insulator plus conduct list form
IFLT	Floating surface list for PWASON
IMAT	Material type of each surface
ISEC	Ion secondary current to each surface
JFIX	Similar to IFIX only for each surface cell
JINE	Incident e^- to SURFS
JINI	Incident ion to each surface
JPRM	List per surface of derivative of total current by surface voltage
JSM	Small surface currents for insulators and conductor list
JSMS	Total of "small" currents to each surface
JTOT	Total surface current list for insulators and conductor list
JTSR	Total current to each surface
KSURF	Surface cell list
LAUX	Coordinate list for auxiliary values
LCEL	Surface to node connectivity list
LCOF	Table to provide direct addressing to the LCEL list by element address
LIST	Defines element location ICCG packed sparse matrix
LTBL	Element table
LTYP	Element location with respect to sheath. Also a calculation saver used by CURREN.
MRKR	Marks the starts of new rows in LIST
MU	Product of matrix M dot U

NLST	A list of insulating surfaces
OLDV	Voltage change from previous charge step
OLJT	Previous CHARGE cycle total surface currents from first trial step
OLSV	Surface voltages used to find OLJT
OPOT	Previous space charge iteration potentials
POT	Node potentials
PSGM	Pre sigma (conductance) calculated by VEHICL and ORIENT. Does not include grid size.
QE	Average element electric field
QUSD	Stabilized charge density array
R	Node residuals
RHOI	Sheath ion densities
RHOE	Sheath electron densities
RHS	Right hand side of charging equation
RMAT	The capacitance matrix portion of charging equation
SCRN	Linearized screening term
SCRT	Scratch vectors
SEDL	Surface edge list
SFAR	Area of each surface
SFHC	Weighted sum of h+ particles which impacted on object during CURREN
SGMA	Conductivity matrix including grid size
SPAC	Work space
SRDT	Keeps the voltage of fixed surfaces
SREL	List of pointers per surface into the LCEL list
SRFC	Weighted sums of particles which impacted on object during particle pushing

SRFE	Pushed e^- currents to surface
SRFH	Pushed h^+ currents to surface
SRFI	List of surface ion currents
SRFR	Surface residual list
SRFU	Surface U list
SRFV	Surface voltage list
SRFY	Surface Y list
SRHA	Estimate of ambient h^+ surface current
SRIA	Estimate of ambient ion surface current
SRTS	Surface unpermutation ordering information
SUMM	Code history information, summary information from previous PWASON, CURREN and CHARGE cycles (not implemented yet)
SUNC	Photo current to each surface
TKSR	Sets CBUF space to hold all KSURFs at once
TSRV	Trial surface voltage list
U	Conjugate gradient error vector
VCHG	Volume charge associated with each surface
VDT	Change in voltage portion of charging equations (the answer found by ICCG)
VMAP	Velocity space map, work space
VMC	Voltage vector used to construct RHS (voltage minus conductor)
VXB	V cross B voltage bias for each insulating surface

5.40 OBJECT DEFINITION

The interpretation of the object definition keywords is done by VEHICL. The building blocks (Section 6.10) are recognized and used to define entries in an element table and entries in a surface cell list. The element table describes the contents of each cube making up the object definition grid. The surface cell list describes the type of surfaces which define the object. POLAR models objects using their surfaces as an inner boundary to the calculation grid. The majority of computations done by POLAR deal with the space surrounding the object. Some parts, i.e., CHARGE, look at changes which occur on this boundary.

After the object has been translated to element and surface descriptions, VEHICL can generate the various lists used by the other modules.

5.50 POTENTIALS

The potential solver, PWASON, finds the space potentials surrounding the object. The algorithm used is discussed in Section 4.31. This discussion is limited to the coding structure. Figure 5.50/1 illustrates the organization of PWASON. The convergence test used for the space charge iteration is to find the RMS error between the two most recent cycles.

CONGRD performs the conjugate gradient calculation test for CGM convergence (4.21.1). Figure 5.50/2 outlines the structure of CONGRD.

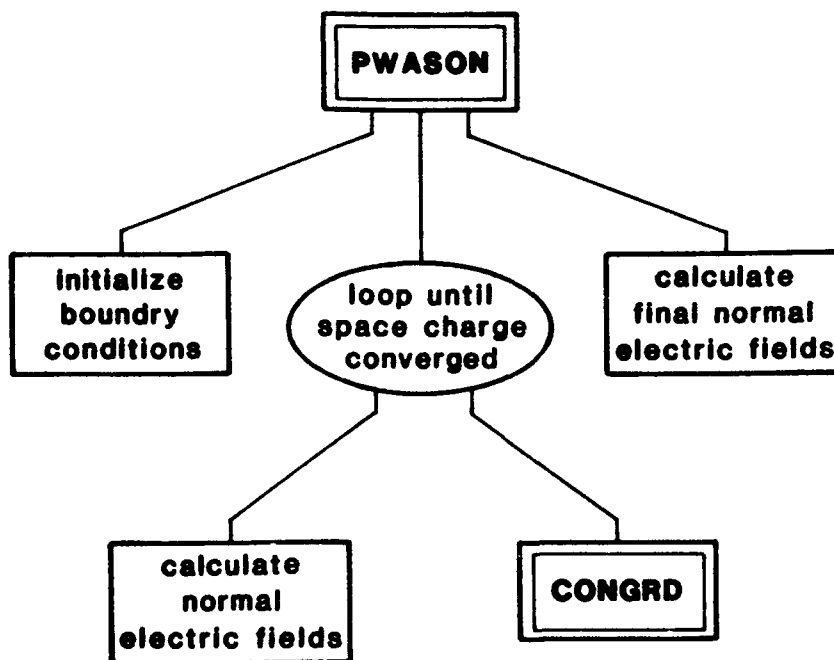


Figure 5.50/1. PWASON structure.

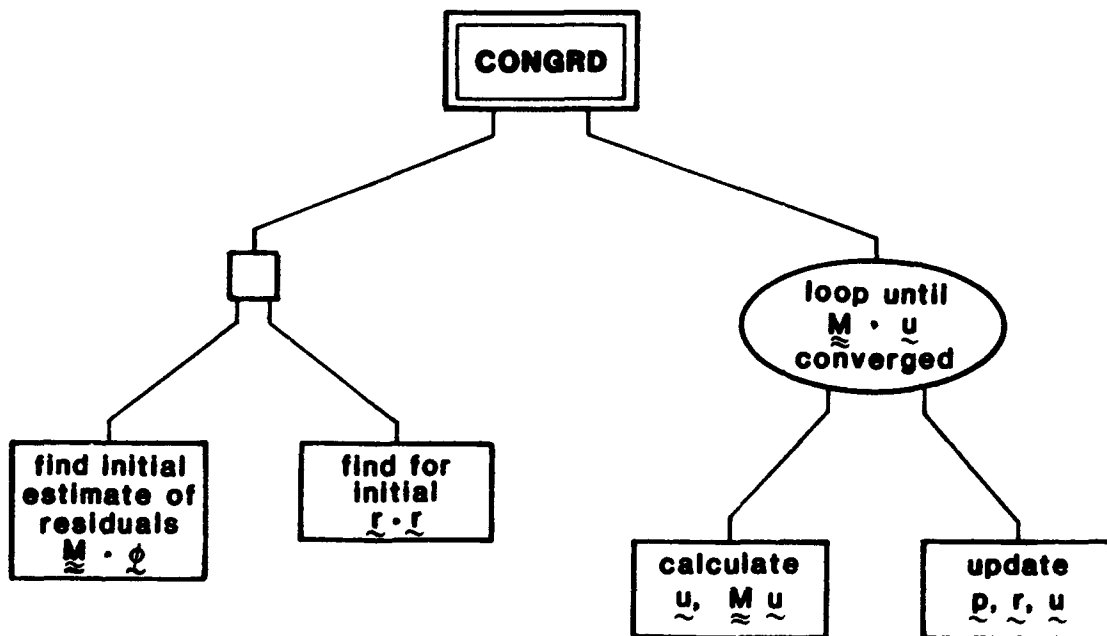


Figure 5.50/2. CONGRD structure.

5.60 PARTICLE DENSITIES

The calculation of charge densities is performed in two major segments. Before any electric field information is available the straight line thermal ion density is calculated everywhere in the grid. When potentials have been calculated, the sheath edge is found and particles are tracked from the sheath edge to the object. Once a sheath has been defined, the straight line thermal ion densities can be recalculated. The algorithms for these two calculations are described in the following sections.

5.61 PRESHEATH SPACE CHARGE DENSITIES

Two algorithms are available to calculate presheath densities. The difference between the approaches lies in the treatment of the spacecraft and sheath geometries. The NEUDEN method creates a more accurate representation of the spacecraft than the SHADO algorithm. The SHADO algorithm is much faster and is able to include the sheath generated wake in its calculations. The presheath electron densities are defined to be of equal magnitude, but of opposite sign, as the ion densities since the presheath densities are defined to be quasi-neutral.

5.61.10 NEUTRAL ION APPROXIMATION (NEUDEN)

The presheath ion density calculation consists of determining the function $g(x, \Omega)$ (described in Section 3.31). This function has value unity when particles can reach point x from angle Ω without hitting the vehicle, and it has value zero when the particles are prevented from reaching x because they are blocked by the object. The major routine in this section of the code is GEMFAC for geometrical factor, and the results used by NTERAK are GI's for geometrical ions. The GI's are element centered and defined as

$$GI(I, J, K) = \frac{1}{n_{oi}} \int g(x, \Omega) \int f_{io}(r, \nu) \nu^2 d\nu d\Omega$$

For calculation of these geometrical factors we work in a polar coordinate system whose $\theta = 0$, ions. In this system, the function f_{i0} is azimuthally symmetric and the integral is reduced to

$$GI(I,J,K) = \sum_{\theta_i} f(\theta_i) \left[\sum_{\phi_i} g(\theta_i, \phi_i) \Delta\phi_i \right] \Delta\theta_i$$

where the function $f(\theta)$ is normalized so that

$$\sum_{\theta_i} \sum_{\phi_i} f(\theta_i) \Delta\theta_i \Delta\phi_i = 1$$

This function is calculated in FCAL. The calculations are done on a uniform grid with a default resolution of 1° in θ and 10° in ϕ . The greater resolution in the polar angle θ is necessary since f changes rapidly as we move away from the ram direction. The geometrical factor is calculated by taking the building block surfaces (A2's from HIDCEL) and projecting them onto the θ, ϕ plane. Regions inside of each surface are then marked as excluded. The major source of error is the interpolation between vertices in θ, ϕ space are not straight lines in , straight line edges in Cartesian space are not straight line edges in Cartesian space are not straight lines in θ, ϕ space. To improve accuracy, ADVERT adds extra vertices in the middle of edges for each A2 surface. The number of vertices per edge is controlled by NADD, and has default value of 2. The calculations to determine the points inside the surfaces are usually done in STICK (for STICK a one into the GEM array). However, the cases of the ram or anti-ram directions being excluded must be treated separately. The poles in the θ, ϕ space are singular points and are handled in STKUP and STKDN for the $\theta = 0$ and $\theta = \pi$ poles, respectively.

GEMFAC is called for each surface by NEUDEN (for neutral approximation density) which also performs the angle summations. The principle advantage of this geometrical technique is speed, the tens of millions of $g(\theta, \phi)$ needed for the densities at each grid point can be calculated in just a few minutes of computer time. A few representative plots of ram, regular and anti-ram polar angle projections are shown in Figures 5.61/1-5.61/3.

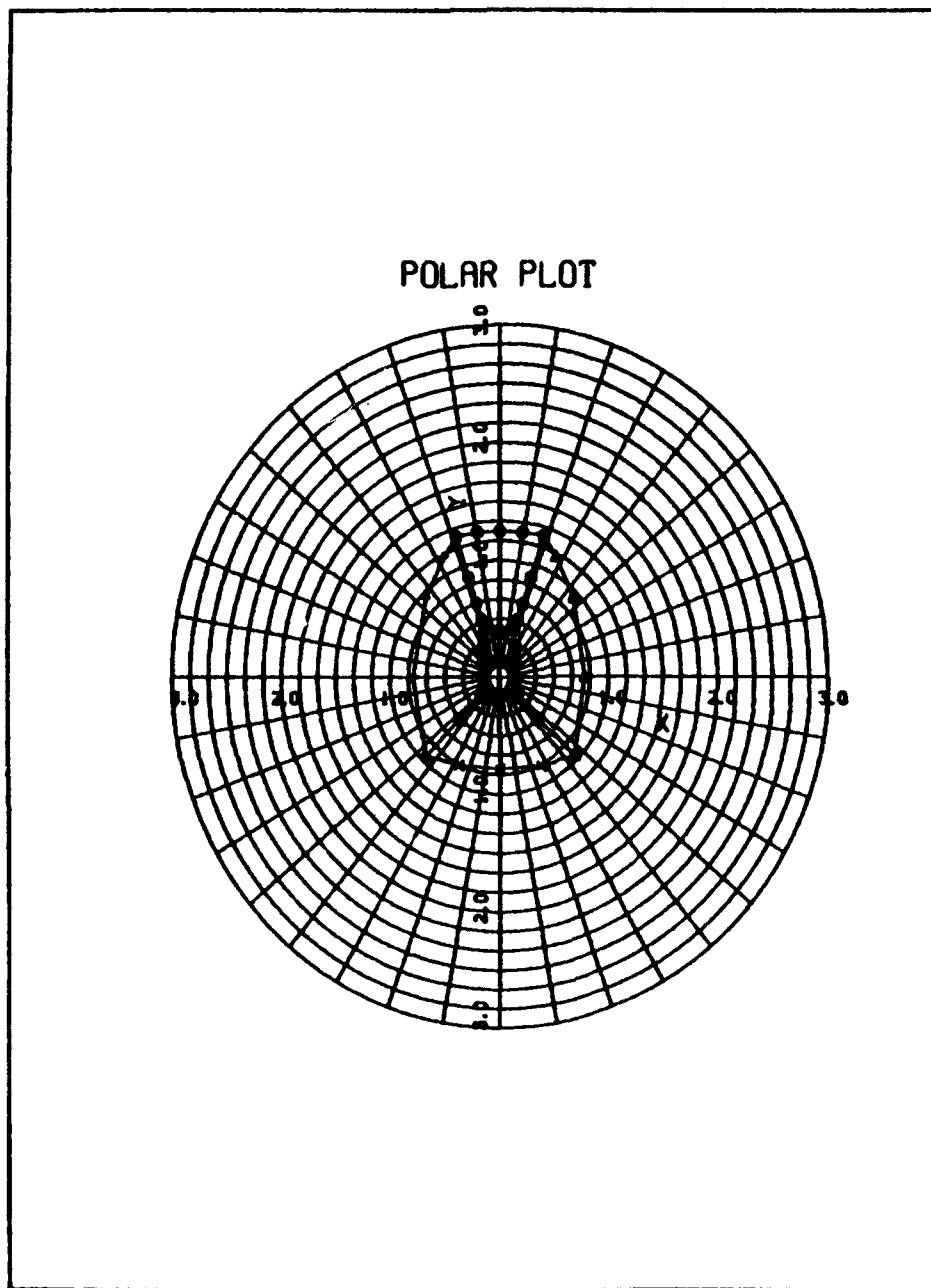


Figure 5.61/1. Neutral approximation phase space map of a flying brick blocking the ram direction.

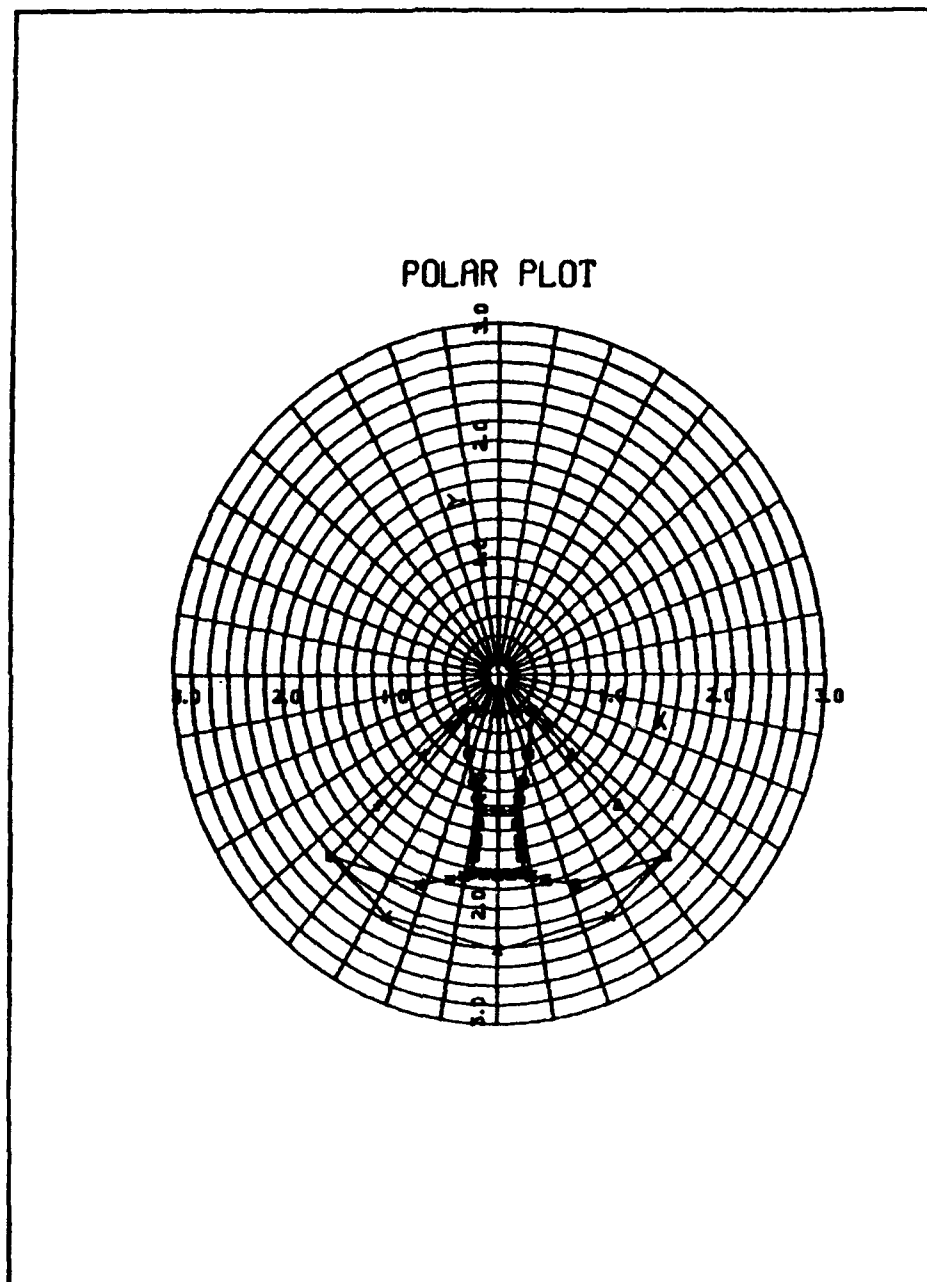


Figure 5.61/2. Neutral approximation phase space map of a flying brick at right angles to the ram direction.

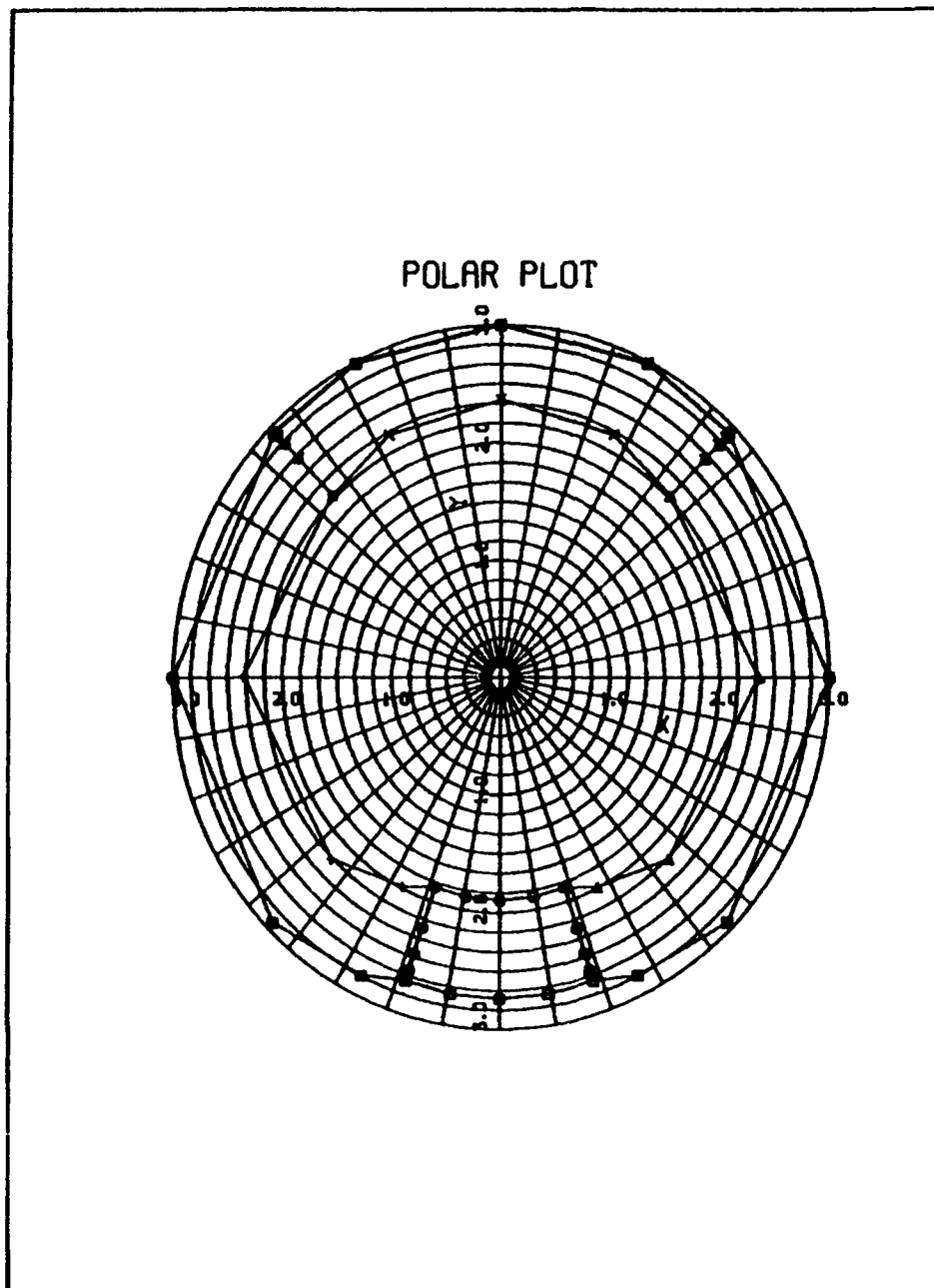


Figure 5.61/3. Flying brick in the anti-ram direction.

5.61.15 SHADO APPROACH

SHADO is a new module for use in computing the particle wake behind an object in Low Earth Orbit (LEO) traveling at mesosonic speeds.

Two simplifying assumptions are made in computing the ion densities in the wake. The first assumption is the neutral ion approximation which assumes ions travel in straight lines and are stopped if they impact the spacecraft. This neglects electric field effects on ion trajectories. The second assumption is that ion filling of the wake is due to electric field acceleration. The neutral ion approximation is expressed by the following equation:

$$f_i(\vec{x}, \vec{v}) = g(\vec{x}, \Omega) f_{io}(\vec{v}) \quad (1)$$

where $f_i(\vec{x}, \vec{v})$ is the ion distribution function at a point \vec{x} in space for a velocity \vec{v} and $f_{io}(\vec{v})$ is the unperturbed velocity distribution function for a drifting Maxwellian. The function $g(\vec{x}, \Omega)$ has a value of zero if a ray starting from \vec{x} in direction Ω would strike the spacecraft; otherwise it is one. The charge density is obtained by integration over velocities:

$$n_i(\vec{x}) = \int f(\vec{x}, \vec{v}) = \int g(\vec{x}, \Omega) \left[\int_0^\infty f_{io}(\vec{v}, \Omega) v^2 dv \right] d\Omega \quad (2)$$

The object being studied is described by a collection of surface elements which are referred to as object definition surface element, but include object and sheath surfaces. In order to determine the charge density at a point in space, it is necessary to determine the solid angle which is occupied by the object and performing the integration in equation (2). The problem is reduced to finding $g(\vec{x}, \Omega)$ given \vec{x} due to the neutral ion approximation.

While the NEUDEN (5.61.10) algorithm is practical for simple objects, it becomes clear that complex objects with many surface elements will be increasingly difficult to handle. This is especially true when several million \vec{x} coordinates are required to adequately compute the particle wake behind a spacecraft. Another application for which this approach is inadequate is problem of sheath shadowing. The sheath around the moving object can be described using surface elements and treated as an object. However, very many surface elements are required and the old algorithm is too cumbersome to compute both the particle wake behind the object and the sheath shadowing.

The SHADO approach was developed to simplify the object definition so that the charge density may be computed much more rapidly. The new algorithm relies heavily on the fact that the ion distribution, relative to the spacecraft, is heavily weighted towards the ram direction at mesosonic velocities. Since it is azimuthally symmetric, equation (2) can be reduced to:

$$GI(\vec{x}) = \sum_{\theta_i} f_{cum}(\theta_i) \left[\sum_{\phi_i} g(\theta_i, \phi_i) \right] \delta\theta_i \quad (3)$$

where the GI's are called 'geometrical ions' obtained by dividing equation (2) by the unperturbed ion density (n_{oi}). If a point \vec{x} is completely blocked by the spacecraft, GI will be zero, while GI will be unity far from the spacecraft.

The function $f_{cum}(\theta)$ is obtained by integrating $f_{io}(\theta, \phi | \vec{v})$ over ϕ since it is azimuthally symmetric, and normalizing it as follows:

$$f_{cum}(\theta) = \sum_{\phi_i} f_{io}(\theta_i, \phi_i, v) \Delta \phi_i \quad (4)$$

$$\sum_{\theta_i} f_{cum}(\theta) \Delta \theta_i = 1 \quad (5)$$

f_{cum} is calculated in advance, given \vec{v} , and is stored internally as an array which rapidly decreases as θ increases for mesosonic velocities. The value of θ at which f_{cum} drops to 0.01 of its value in the ram direction ($\theta = 0$) is computed and saved as $\theta_{99\%}$. As $|\vec{v}|$ increases, $\theta_{99\%}$ decreases indicating that the Maxwellian distribution of ions is shifted towards the ram direction. Surfaces lying within $0 \leq \theta \leq \theta_{99\%}$ from the perspective of \vec{x} will therefore have a significant effect on the value of $GI(\vec{x})$, while surfaces positioned beyond $\theta_{99\%}$ will have a negligible impact.

Simplification of the object definition is achieved by first projecting the object onto a surface which is normal to the ram direction and then overlaying this projection with a new two dimensional grid. The nodes on this grid are arranged in an equilaterally spaced staggered array resulting in hexagonal elements. These hexagonal elements provide greater angular resolution of the object projection than an equivalent cartesian grid system. Each surface element in the object definition is then sequentially projected onto this hexagonal grid and a depth is computed for each grid node which falls inside the surface element projection.

This depth represents the distance from the projection surface to the surface element along a ray parallel to the ram direction. The projection surface is placed so that the forward-most point on the object corresponds to zero depth. All of these calculations are performed only once prior to evaluation of any GI's.

The number of nodes making up the hexagonal mesh is controlled by a parameter m_{hex} . The nodes are distributed uniformly over the rectangular area on the projection plane which just encompasses the entire object projection. As each object definition surface is processed, a table of depths for each node in the hexagonal mesh is maintained. After all the surfaces are projected, the table for each node is sorted by increasing depth. Thus, at each node on the hexagonal grid there is a table of depths representing the intersection of a ray beginning at the node and any of the object definition surfaces in the path along the anti-ram direction. Given a \vec{x} coordinate, it is then simple to compute its distance from the projection plane and its location on the hexagonal grid. Comparing the depth of \vec{x} to the table of depths of the adjacent hexagonal grid nodes it is quickly determined whether \vec{x} is in front of, behind, inside, or adjacent to the object.

If the reference point, \vec{x} , is in front of the object, then GI (\vec{x}) will be unity because all angles θ pointing to the object definition surfaces will be much larger than $\theta_{99\%}$. If the reference point is sufficiently far away from the object such that the entire object lies outside of the $\theta_{99\%}$ limit, then GI is set to unity as well. If the reference point is inside the object, then GI is set to zero. Since many of the points in the object mesh fall into one of these categories, little computational effort is expended in determining the GI (\vec{x})'s. For the remaining points in the object mesh, it is necessary to determine where the object is relative to \vec{x} and determine how much of the view towards the ram direction is blocked.

Much of the computational speed improvement in the SHADO module is attributable to the simplification of the object representation already

described. Given an \vec{x} coordinate, say directly behind the object, a scan along a number of rays controlled by a parameter N_{phi} , is initiated. Each ray begins at \vec{x} and proceeds along a fixed angle ϕ in the plane of the projection surface. At intervals just less than the hexagonal mesh element size, the surrounding hexagonal nodes are tested to determine whether they are over the object or not. If the table of depths for one or more of these nodes is empty, then this indicates that the scan has proceeded beyond the projection of the object. A quick calculation is made which approximates the distance to the object outline (accurate to half the hexagonal mesh dimension) and the depth to the object at this point is computed. This gives the angle θ from \vec{x} to the boundary of the object and the contribution to $GI(\vec{x})$ along this value of θ is determined. Summing for each of the N_{phi} values of θ gives the value the geometrical ion at \vec{x} .

A number of variations on the scheme outlined above are possible based on whether the point \vec{x} is directly behind (the table of depths on all nodes adjacent to \vec{x} are non-zero), or off to one side of the object. The same basic procedure is followed however, by scanning in rays along the projection plane, determining where the object is relative to \vec{x} on this plane, and then determining the angles, θ , locating the boundaries of the object in three dimensions relative to \vec{x} . The accuracy of the object description using this algorithm is controlled by the number of hexagonal grid nodes (N_{hex}) placed on the hexagonal grid and the number of scan angles (N_{phi}) used when locating the object relative to \vec{x} .

5.61.16 SHADO STRUCTURE

The SHADO routines are organized in a highly modular fashion. The following sections detail the organization of the routines and describes their function. The overall structure diagram for SHADO is shown in Figure 5.61/4. The driver is the POLAR code, SHDSET is the routine which processes the object definition surface elements, and SHADO is the routine which computes the GI given \vec{x} .

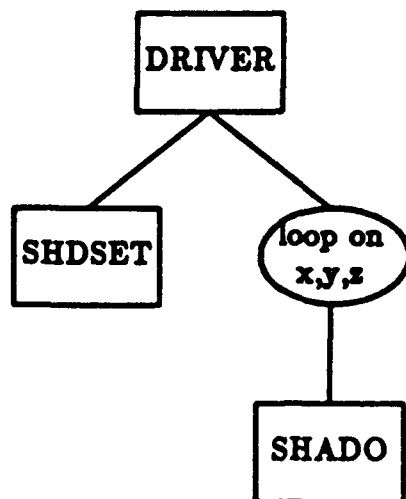


Figure 5.61/4: Shado Structure Diagram

SHDSET - Set Up Routine

Initialization for the SHADO routine takes place in SHDSET and consists of looping over each surface defining an object, projecting that surface onto the plane perpendicular to the ram direction, and determining which nodes on a hexagonal grid are shaded by each surface. For each surface, limits are determined which define a box in the hex mesh which encloses the projected surface. Only points inside this box are checked for shadowing by that surface.

At each mesh point in the hexagonal grid, a record is kept of the height of intersection from the projection surface and the number of surfaces which shade a given mesh point. An array stores the z values of each surface intersection sorted by ascending node number and then by ascending z value. Given any node, it is then simple to determine if this node is shaded in the ram direction and to find the entry and exit z values.

The structure diagram for SHDSET is shown in Figure 5.61/5.

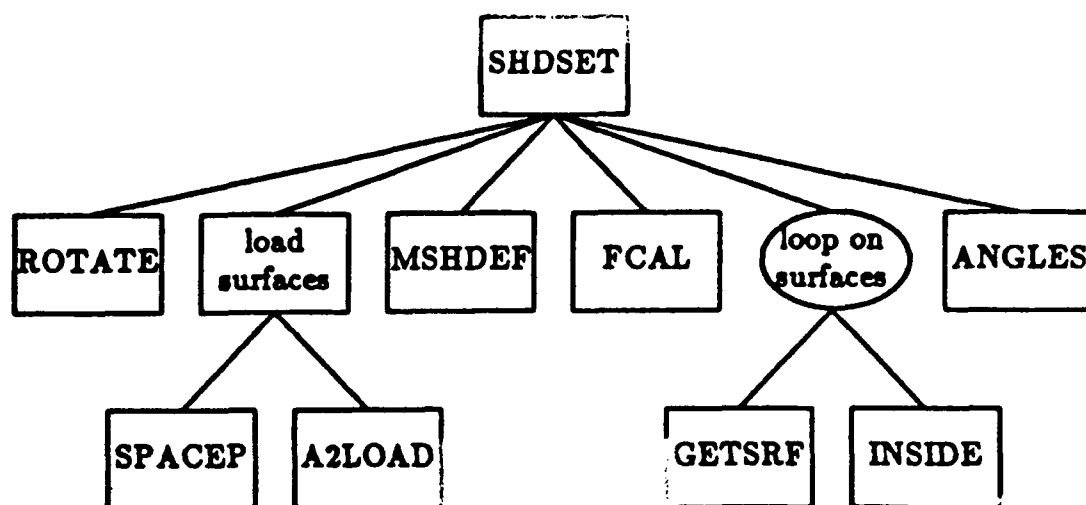


Figure 5.61/5: SHDSET structure diagram

ROTATE

Computes the rotation matrix required to convert any (x,y,z) to (x',y',z') in which the $x'-y'$ plane is normal to the Mach vector. The x axis is taken to be in the $x-z$ plane.

SPACEP A2LOAD

Reads in surface element data for the National aerospace plane using the MSIO library. Alternatively, A2 surfaces may be read using routines A2LOAD and A2PREP. These routines simply read in data from MSIO data files and load an array of vertices for each surface. These routines will also find the minimum and maximum x,y , and z values within the object to be used in defining the hexagonal mesh. Data selection is determined by the LUSURF parameter which corresponds to the logical unit number for the surface data; 19 for the A2's, 9 for the spaceplane data. Surfaces are defined by coordinates for surface vertices and an outward pointing normal vector. This form is exactly how A2 surfaces are defined, for the spaceplane, however, the normal is computed using three distinct vertices to compute the normal vector assuming clockwise node numbering.

MSHDEF

MSHDEF fits the projection of the shadowing object (either a spacecraft and/or the plasma sheath) with a 2-dimensional hex grid. The constraints are to resolve the object's edge as well as possible while using less than the maximum number of hex mesh points. To do this, hex mesh lengths are guessed and then the mesh is checked to see how close the grid boundary is to the object.

Using the minimum and maximum x, y , and z values, a box is defined within which the object is fully contained. The eight vertices making up this box are then projected onto the $x'-y'$ plane and the minimum and maximum x' and y' values are found. A hexagonal mesh is then established which has its origin at $(x_{\min} - \Delta, y_{\min} - \Delta)$ and extends up to $(x_{\max} + \Delta, y_{\max} + \Delta)$ where Δ is the mesh spacing. Δ is chosen using iteration such that the number of mesh points is $\leq N_{\text{hex}}$. The mesh is then skewed 60 degrees to give a square mesh and normalized by Δ such that the nodes on the mesh are integer values.

FCAL

Calculates the ion density factor array for a point in space which is unobstructed in all directions. The ion density factor is computed as a function of the out of plane angle θ as follows:

$$f(\theta) = \frac{2Ae^{-y^2}}{\sqrt{\pi}} \quad (6)$$

Where:

$$y = |\vec{V}_{\text{mach}}| \frac{\cos \theta}{\sqrt{2}} \quad (7)$$

$$A = \begin{cases} \sqrt{\pi} e^{y^2} (1 - \text{erf}(y)) & y \leq 3 \\ y^{-1} (1 - \frac{1}{2}y^{-2} + \frac{3}{4}y^{-4} - \frac{15}{8}y^{-12} + \frac{105}{16}y^{-16}) & y > 3 \end{cases} \quad (8)$$

The $f(\theta)$'s are calculated over a uniform grid with a resolution of 2 degrees in θ and then integrated over θ to give:

$$f_{\text{cum}}(\theta) = \int_0^\theta f(\gamma) d\gamma \quad (9)$$

GETSRF

Reads in the surfaces of the object and returns the internal limits for the hexagonal mesh points which must be checked for shadowing. Any surface which is perpendicular to the projection surface is ignored. A loop inside SHDSET reads in the number of surfaces from this file, and GETSRF gets a valid surface. The return arguments are the coordinates of up to four vertices making up a surface element which is not perpendicular to the x', y' plane.

INSIDE

Fills an array which retains all of the surface intersections at each hexagonal grid point which is shaded by a surface. For each mesh point which is blocked by the given surface, the height of the surface above the projection surface is computed and stored in an array of z 's and the corresponding node number is stored in a companion array. After all of the surfaces have been processed, these two arrays are sorted by node in ascending order of z' to obtain the minimum z value (z_{entry}) and maximum z value (z_{exit}) for each of the blocked nodes.

ANGLES

In SHADO, the hexagonal mesh is skewed by 60 degrees to become a cartesian grid system. Scanning to define the object boundary takes place over a given number of angles in the $x'-y'$ plane, which when skewed, become lines which are no longer uniformly spaced angularly. This routine computes the increments in x' and y' to be taken along each of these lines. The increments are selected to be 99 unskewed mesh:

$$\Delta x'_i = 0.99(\sqrt{3}\sin\phi_i - \cos\phi_i) \quad (10)$$

$$\Delta y'_i = 0.99 \cos\phi_i \quad (11)$$

Where ϕ is the angle in the x', y' plane measured from the y' axis and incremented by: $\phi_i = \left[i - \frac{1}{2}\right] \frac{\pi}{N_\phi}$; N_ϕ is the number of scan angles to be used in the x', y' plane.

SHADO - Calculate Geometrical Ion Density

SHADO calculates the presheath ion density using the neutral ion approximation. This approximation assumes that ion motion is perturbed only by collisions with an absorbing object. The ion density accounts fully for ion thermal velocities, but neglects trajectory bending by electric or magnetic fields. The ion density at a point is obtained by integrating the ion distribution at that point over the range of ion velocities:

$$n_i(\vec{x}) = \int f(\vec{x}, \vec{v}) d\vec{v} \quad (12)$$

The ion distribution is a function of position \vec{x} and of the ion velocities \vec{v} . The ion distribution function is redefined as the product of the unperturbed velocity distribution function and a geometrical factor as follows:

$$f(\vec{x}, \vec{v}) = \sum_{\phi_i=1}^{N_\phi} g(\phi_i) f_{\text{cum}}(\theta) / N_\phi \quad (13)$$

The $f_{\text{cum}}(\theta)$ array was computed in FCAL while the geometrical factor is determined by scanning along lines of constant ϕ in the x', y' plane and finding the points at which an unblocked position becomes blocked or a blocked position becomes unblocked. If the given point is behind the object, a θ can be found by interpolating for the object boundary between blocked and unblocked scans and finding the z_{exit} for this interpolated position. The angle $\theta = \arctan\left(\frac{r}{z_{\text{exit}}}\right)$ where r is the distance from the interpolated boundary position to the given point.

The structure diagram for SHADO is shown in Figure 5.61/6.

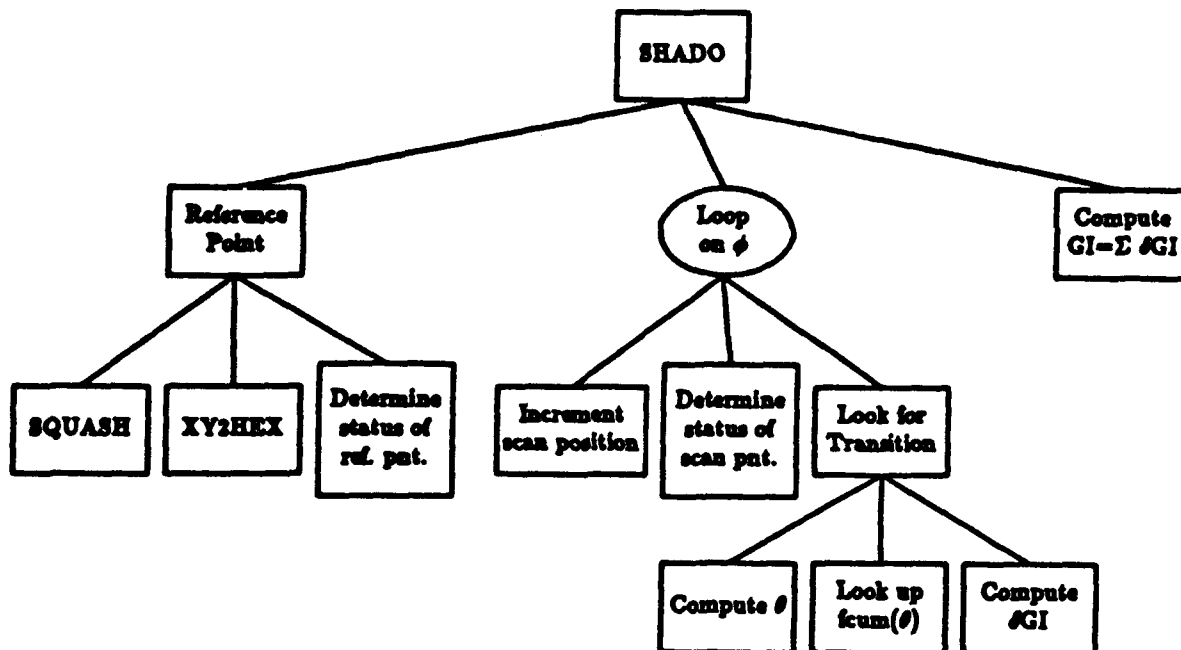


Figure 5.61/6: Structure Diagram for the SHADO subroutine

The majority of coding in SHADO is devoted to determining the status of the reference point and the scan points on the hexagonal projection mesh. The status includes determinations of whether a particular point is within the hexagonal mesh or off of it; whether it is in front of the object; whether the point is blocked in the ram direction, and if blocked, whether the point is inside the object. As indicated in the structure diagram, the reference point (input) is first checked. If the point is inside the object, in front of it, or so far off to the side of the object that it would not be possible to intersect the object within the 99.9 limit, then the GI value is returned as zero without further calculation.

Once the reference point status is determined, a loop is entered which initiates scans along the scan angles θ_i calculated in ANGLES. For each θ_i , a transition is sought which marks the outline of the object on the projection surface. If the reference point is not blocked with respect to the ram direction, then a transition is sought where the object first blocks a scan point. If the reference point is blocked, then a transition to unblocked status is sought.

When a transition is found, the angle θ is computed which is the angle from the reference point to the object surface at the transition point. If the reference point is blocked and behind the object, the density contribution for θ_i is $f_{\text{cum}}(\theta)/N_\theta$. If the reference point is unblocked, then density contribution is $1.-f_{\text{cum}}(\theta_1) + f_{\text{cum}}(\theta_2)$ where θ_1 is the angular position of the first transition to blocked status, and θ_2 is the transition back to unblocked status.

5.61.20 ELECTRIC FIELD CORRECTION FOR NEUTRAL IONS

An optional multiplying factor has been included in the ion density calculation of NTERAK to correct for weak electric field focusing and the expansion front seen in the deep wake. The correction was implemented in the following manner:

A variable length table of correction factors is produced along with the associated ion densities during initialization by SETEFC which is called by NEUSET.

SETEFC calculates the ion densities and their corrections for an infinite half plane. These values are calculated using the following set of equations:

For $\theta = 0$ to π

Let

$$Y = \left| \vec{V}_{\text{Mach}} \right| \frac{\cos \theta}{\sqrt{2}}$$

and let

$$A = \begin{cases} \sqrt{\pi} e^{y^2} (1 - \text{erf}(y)) & y \leq z \\ y^{-1} \left(1 - \frac{1}{2} y^{-2} + \frac{3}{4} y^{-4} - \frac{15}{8} y^{-12} + \frac{105}{16} y^{-16} \right) & y > z \end{cases}$$

where

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-u^2} du$$

Using these variables, the ion density before correction as a function of θ is

$$F(\theta) = \frac{A}{\sqrt{2}} e^{-y^2}$$

The correction factor, $C(\theta)$, for the electric field effect is found by the following series of operations:

For

$$\theta > \tan^{-1} \left(\frac{-\log(0.7)}{\sqrt{2} |\vec{V}_{\text{Mach}}|} \right) + \frac{\pi}{2}$$

$$C(\theta) = 1.0$$

and for smaller θ ,

$$C(\theta) = \frac{0.7 e^{\sqrt{2} |\vec{V}_{\text{Mach}}| \tan(\theta - \pi/2)}}{F(\theta)}$$

This $C(\theta)$ is redefined

$$\text{if } C(\theta) < 1.0 \text{ then } C(\theta) = 1.0$$

and then

$$\text{if } C(\theta) F(\theta) > 1.0 \text{ } C(\theta) = 1/F(\theta)$$

NEUDEN uses the tables created by SETEFC by calling the routine EFCORV to find the corrected g_i (ion density) for an initial ion density.

EFCORV checks for the special case of $|\vec{V}_{\text{Mach}}| = 0$ (no correction) before finding the correction factor. If this is not the case, EFCORV looks for the range of entries in the $F(\theta)$ which includes

the neutral wake density sent to it by the calling routine, using a binary search algorithm. The search is sped up by first checking the interval which included the ion density found in the previous call to EFCORV.

When the correct range is found, θ is found by linear interpolation. This is then used to find $C(\theta)$, also using linear interpolation. The corrected ion density is then found by multiplying the correction factor and the neutral wake density.

5.62 SHEATH PARTICLES

The space charge density of the attracted specie internal to the sheath boundary, and currents of the particles to the vehicle surface must be determined by particle tracking. The concepts involved were discussed in Section 3.32 and 4.42. This section documents the coding responsible for these calculations, but a brief overview is in order.

External to the sheath, electric fields are weak enough to allow for accurate estimates of the flux to any portion of the sheath surface, including ram effects for ions. Presuming a previous Poisson calculation, the sheath is currently defined to be an equipotential near the ambient plasma temperature (this is an input parameter). Once the sheath is located, it is divided into subareas. These subareas subsequently become 'particles' which represent a constant current, rather than constant charge. This current, referred to as current weights, is the subarea times the input flux density of ions (Section 4.42) to the subarea. The actual particle tracking is done per slice (4.11) with the particle 'pusher' sweeping the grid alternately in the +Z (called right) and -Z (left) direction. Trajectories evolve in the X and Y coordinate directions until they exit a slice whereupon the trajectory information is stored. The trajectory is picked up in the next slice or the return pass of the pusher. Ultimately, they leave the problem or more commonly, hit a surface and are moved to a 'dead-particle' list, and are later processed into surface currents.

Throughout all of this pushing, the time a particle spends in a volume element is multiplied by its current to determine the space charge contribution to the element.

5.62.10 SHEATH EDGE

The controlling routine for the sheath edge and particle assignment algorithms is STHCAL. STHCAL uses two boundary potentials, CURPOT and BNDPOT. These are either defaulted in OPTDEF, input through OPTIN, or calculated by SETENV depending on user and model requirements. CURPOT is used to calculate the presheath to sheath fluxes. This is done only once for NTABLE different angles between the sheath edge normal and flow vector. Individual sheath subareas (corresponding 1 to 1 with particles) are assigned fluxes by interpolation. BNDPOT is the potential of the sheath edge and is usually the same as CURPOT, but is sometimes set to PSIM (4.44.2, 3.60) or a user input value.

Once BNDPOT is known, STHCAL loops through all of the elements, calling SHEATH to check each element to see if it contains the sheath potential. If part of a boundary passes through a cell, it divides the boundary surface into triangular pieces. These pieces represent potential particles which are assigned a current equal to the piece area times the sheath flux for that location and normal (interpolated by the function STHWGT). Their initial position is the centroid of the triangle and their initial velocity is found using BNDPOT and the method described in Section 4.42.2.

SHEATH will occasionally find particles that should be eliminated. This might happen when the sheath edge equipotential lies between two regions of opposite polarity, different extensions of a complex object, or otherwise cannot "see infinity". This elimination is affected by DBLCHK which backtracks the trajectory through about three elements to look for non-source regions. Statistics from this process are available in the output by setting KDIAGS(4) to 2 or more (Section 6.44.20).

There are two user-specified options to the process of creating a particle list. The two corresponding keywords are AVEPRTCL for averaging particles and THRMSPRD for thermal spreading particles.

In the AVEPRTCL option, all particles of the same type (ion+, h+, or e-) found at cell (ix,iy,iz) are lumped into a single particle called an average particle which will have the following properties: flux is the total fluxes of all contributing particles; initial position and initial velocity are that of the particle which has the greatest flux; initial energy is calculated with respect to its new initial position and velocity. Particle type and QE are the common particle type and common QE, respectively. All this is done by the routine AVRAGR.

After the particle averaging, if it was requested, each particle is broken into a number of smaller particle with different initial velocities. The method used to thermally spread the sheath particles is described in Section 3.43.

Once all of the particles in a Z-slice (4.11) are defined, they are written onto file 10 with their current, initial position, initial velocity, and initial total energy. Then the particle lists are stored in a linked list data structure (5.62.11) in chunks of 100 or fewer particles.

5.62.11 THE PARTICLE LIST STRUCTURE

The particle list is kept in a linked list data structure. An array in the common block /LNKCOM/ contains either -1 or the key of a particle list. The key is key number of the particle list in file 10 as it is used by the MSI0 package.

To find the key of the first particle list of a certain z-slice, the /LNKCOM/ array LKPART(i) (LKSCUR(i) for surface currents) is indexed by the z-coordinate of the slice (4.11, 4.12). The rest of the particle lists for a given z-coordinate are indexed by the first word from the preceding list. This is the MSI0 key of the next slice. When a particle list does not have another list following it, the key it holds is a -1. The negative one is a flag signalling the end of a linked list.

For example, if our problem had a z-coordinate which varied from 0 to 6 and 200 particles in z-slice 0, 730 particles in slice 3 and 1300 particles in slice 4, file 10 would be similar to table 5.62.11/1. Since the order in which the particles were stored affects the actual contents of the table, there are a number of possible arrangements.

In the table, 230 particles of the $z = 3$ slice are in file 10 at key 1 and 500 are at key 5. Since next key is -1 for key 5, we know there are no more particles in this slice.

As particles are moved to other slices, space opens up in the middle of the file (keys 3, 4 and 8 in the example). To conserve space and to keep the file packed, the emptied keys are also linked. So when a new particle list needs to be stored on file 10, an emptied key will be used if there are any available. The key of the first empty location is stored in the variable IOLDKY.

TABLE 5.02.11/1
AN EXAMPLE OF THE PARTICLE LIST DATA STRUCTURE

First Key List:

		LKPART(i)							
	<u>IOLDKY</u>	<u>i = 0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	
key =	4	6	-1	-1	1	2	-1	-1	

file 10 contents

Key No.	Next Key	No. Particles	z-slice
1	5	230	3
2	9	300	4
3	-1	0	-
4	8	0	-
5	-1	500	3
6	-1	200	0
7	-1	500	-
8	3	0	-
9	7	500	4
10	empty	empty	-

Note: This example assumes a maximum page size of 500 particles.

To store the surface currents, another array LKSCUR(i) is used to hold the key of the first particle list for each z-slice. The linked lists allow the surface currents to use the empty spaces left by the particle list used by the sheath particle tracking section, which helps keep the file packed.

5.62.12 PARTICLE PUSHING UNITS

It is convenient to define special units for the trajectory calculations. The time is in units of the ion acoustic period, the distance is in grid lengths, the velocities are in Mach velocities, and the acceleration is unitless. The quantities are defined by

$$t(\text{code}) = t(\text{sec}) * \frac{v_s (\text{ion acoustic speed})}{h (\text{grid length in meters})}$$

$$x(\text{code}) = x(\text{meters})/h(\text{grid spacing in meters})$$

$$v(\text{code}) = v(\text{meters/sec})/v_s (\text{ion acoustic speed})$$

$$a(\text{code}) = QE = \frac{E_v (\text{electric field in volts})}{T_v (\text{ion temperature in volts})}$$

where the ion acoustic speed is

$$v_s = \sqrt{\frac{kT}{m}} \quad k\text{-Boltzman, } T \text{ } ^\circ\text{K, } m - \text{kg}$$

For the particle currents we have

$$I(\text{code}) = A(x^2(\text{code})) \cdot J(\text{code})$$

$$J(\text{code}) = \frac{J(\#/\text{m}^2 \text{ sec})}{\text{FNORM} \cdot e \cdot \sqrt{2\pi}} = \frac{J}{N \cdot e \cdot V_s}$$

where N is specie density $m (\#/\text{m}^3)$, and e is the electron charge.

For electrons, all of the above expressions are the same except the ion mass is replaced by the electron mass.

5.62.20 SHEATH CURRENT

The routine CURREN controls the process of finding the sheath current (see Figure 5.62.20/1). CURREN calls the initialization and exit routines CURPEP and CUEXIT, respectively, and it pushes the particles alternately to the right (+Z) then left (-Z) until all of the particles have left the grid or hit the object or the push limit, IPCNT, is reached (1 left + 1 right = 1 IPCNT). The variable, NPRTCL, is the number of particles left which have not been pushed to completion.

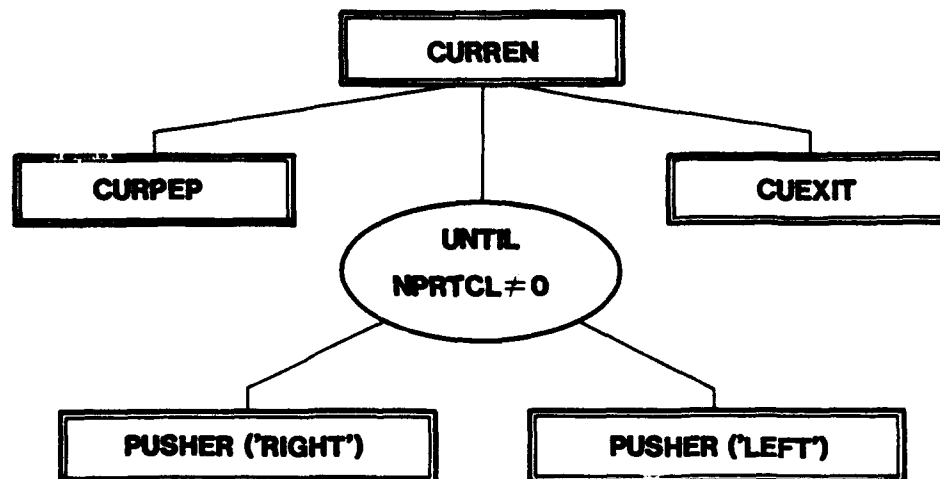


Figure 5.62.20/1. Structure diagram of the subroutine CURREN.

5.62.21 CURPEP (CURRENT PREPARER)

Called by CURREN, CURPEP initializes the particle counters, opens file 10 and initializes it (5.62.11), clears CBUF with a call to the routine BUFCLR, and calls STHCAL to find the initial set of particles.

Once the initial particle lists are calculated, CURPEP resets CBUF for PUSHER with calls to BUFCLR and BUFSET. Then the force table (QE), the extended element table (LTYP), and the sheath current table (RH0I and RH0E) are initialized to complete the routine. To speed up the calculations in XITCEL (5.62.22), the force table (QE) contains the acceleration. The acceleration is equal to the negative trilinear electric field divided by the temperature in volts. The units of the acceleration are grid units (see 5.62.12).

5.62.22 PUSHER (PARTICLE PUSHING)

This routine pushes all the particles to the left or to the right. PUSHER loops through all of the particles in each z-slice, pushing particles until they leave that slice. The particles leaving in the forward direction are pushed again in the next slice and so on. When a particle's path intersects an object face, the particle is transferred to the surface current list (5.62.11) and counted as a dead particle. Particles detected leaving the computational grid are counted and removed from the problem (although their previous trajectory information remains in the RHOI and RHOE lists).

The routine which PUSHER calls to push a particle through a cell is named PUSH.

PUSH decides which of the pushing techniques to use to push a particle out of its cell. PUSH checks the element table (5.23) to see if the element is next to the object or in a partially filled cell (both are labeled 'NEAR' in the LTYP table in CBUF). The routine STPPSH moves ions which are 'NEAR' the object. If the ion is not close to the object, PUSH calls XITCEL (exit cell) to find the time the particle needs to exit the cell, and MOVER to move the particle to its new location.

Similarly, electron particles use ESTEP and EMOVE in place of STPPSH and MOVER, respectively. The electrons are checked when magnetic fields are present to see if the cyclotron radius is larger than the grid size. If it is, ESTEP is used regardless of how close the particle is to the object. When the cyclotron radius is smaller than a half of the mesh size ($dx_{mesh} * \sqrt{RD_{MAX2}}$ See Section 4.52.7), as determined by CHKRAD, a drift approximation is used. In this case the particle's initial velocity is replaced by a drift velocity, $\hat{E} \times \hat{B}$, normal to \hat{B} with the component of velocity parallel to B unchanged. The electric

field is replaced by its parallel component to \vec{B} . The motion is then computed using EMOVE and EXITCL in a manner similar to MOVER and XITCEL or ions.

XITCEL solves the following six quadratic equations for the time, t_{ji} , required for the trajectory to be traced from an entry coordinate X_{oi} to an exit coordinate X_{ji}

$$X_{ji} = X_{oi} + V_{oi}t_{ji} + \frac{1}{2}QE_i t_{ji}^2 \quad \begin{cases} i = X, Y, Z \\ j = +, - \end{cases}$$

where V_{oi} is initial velocity. QE_i is the acceleration calculated from the electric field at the center of the element. The smallest positive real t_{ji} is chosen as the time required by the particle to leave the cell. See Section 5.62.12 for a discussion of the special units used for the particle pushing.

MOVER used the time found by XITCEL to find the new particle location. First it calculates the new position and velocity using

$$X_i = X_{oi} + V_{oi}t + \frac{1}{2}QE_i t^2$$

$$V_i = V_{oi} + QE_i t$$

Then it shifts the particle position by one thousandth of the velocity, moving forward along the exit axis, found by XITCEL, and backwards along the other two. This is necessary because the particle's cell number is referenced by taking the integer portion of the particle position. If a particle stopped exactly on the top side of element (i,j,k) as it moved downward, the address found by truncation would be (i,j,k+1). To prevent this form of faulty addressing, the particle is moved off of the cell boundaries.

Because the central electric field is used for the entire element, and the total energy may drift with each move, MOVER renormalizes the velocity vector to force the particle to conserve energy. After MOVER pushes the particle out of the element, PUSH increments the RHOI (space current density) by the time spent in the cell multiplied by the area of the particle. If the pushed particle was an electron, RHOE is used in place of RHOI.

When the particle gets closer to the object, MOVER can no longer be used since the particle could strike an object surface before it left the cell. To push particles within a grid length of the spacecraft, PUSH calls STPPSH (step push). STPPSH checks to see if the particle is inside a filled portion of an element. If it is, the routine finds which surface it passed through and returns to PUSH. If not, STPPSH calls STPPAR (step particle) which moves a particle for time t . Time t is the smaller of the time required for the particle to free fall or move at a constant velocity a distance of one-tenth of a grid length.

$$t = \text{smaller of } \left[D/|\vec{V}|, \sqrt{\frac{2D}{(|\vec{E}|/\theta)}} \right]$$

where \vec{E} is the electric field at the particles position, θ is the plasma temperature and D is one-tenth of a grid length. When the time step has been computed, the particle is moved to X_i , ($i = x, y, z$)

$$X_i = X_{oi} + V_{oi}t + \frac{1}{2} \frac{E_i}{\theta} t^2$$

$$V_i = V_{oi} + (E_i/\theta)t$$

After the particle is moved, STPPSH increments the RHOI or RHOE (space current density) by the current weight of the particle times the time it was moved. Then checks to see if the particle left the cell by counting how many of the element coordinates of the new position differ from the old. If the particle did not leave the cell, energy

conservation is checked and the velocity is renormalized if there is more than a *5 percent error. Then the particle is pushed again.

If the particle did leave the cell, energy conservation and the number of element coordinates which have changed are checked. If the number is greater than one or the energy is off, the particle is backed up until only one element boundary is crossed and the energy is right. The routine that does this is called MOVBAK (move back). It is important to keep particles from crossing more than one boundary at a time, since it is possible for particles to miss corners of objects.

MOVBAK backs up a particle by finding the average velocity of a particle during the timestep ($\vec{v}_{ave} = (\vec{v}_0 + \vec{v})/2$) then solving $\vec{x} = \vec{x}_0 + \vec{v}_{ave} t$ for t on all of the sides of a cube. Using the smallest positive time it finds, it moves the particle from its original point to \vec{x}' .

$$\vec{x}' = \vec{x}_0 + \vec{v}_{ave} t_{min}$$

MOVBAK then moves the particle off of the cell boundary by shifting the position by 0.001 times the velocity in the same manner as MOVER did previously.

After calling MOVBAK, STPPSH checks energy conservation again. After renormalizing the velocity, if necessary, and incrementing RHODI (or RHODE), the particle is checked to see if it hit a surface as it left the element.

Magnetic field effects for ions are modeled by rotating the velocity vector, at the end of a push, around the magnetic field (see Section 4.42.4).

5.62.23 SHEATH PARTICLE DENSITY

The sheath particle densities, known as RHOI's and RHOE's, are incremented continuously throughout the pushing process. PUSH calls either the pair XITCEL and MOVER, or STPPSH to move a particle, I, across a cell and to calculate the time, t, spent in the cell. As discussed in Section 4.42.5, the density contribution of a particle is just the particle current multiplied by the time spent in a cell divided by the volume (one in these units). The density is incremented

$$\text{RHOI}(x,y,z) = \text{RHOI}(x,y,z) + \text{CURRENT}(I)*t(I)$$

in NUTERM if the push was performed by MOVER, or directly in STPPSH. The above is true for both ions and electrons. (RHOE is used for electrons in place of RHOI).

5.62.24 CUEXIT (CURRENT EXIT ROUTINE)

This routine takes care of the exit from CURREN. It prints a tally of what happened to the particles from SHEDGE and it closes file 10, which now contains the surface currents.

5.70 SURFACE CHARGING

The CHARGE module calculates the change in surface potentials. The attracted species may be modelled in several ways which are described in section 5.72. The default is to use pushed particle densities for the attracted particle surface currents (see section 5.71).

5.71 PUSHED PARTICLE SURFACE CURRENTS

The ion and electron surface currents are found in the lists, SRFI, SRFH, and SRFE, respectively. These lists are created in the segment headed by the subroutine IONCUR from the dead particle list (dead-list) created by the CURREN segment. The upper structure of this segment is illustrated in Figure 5.71/1.

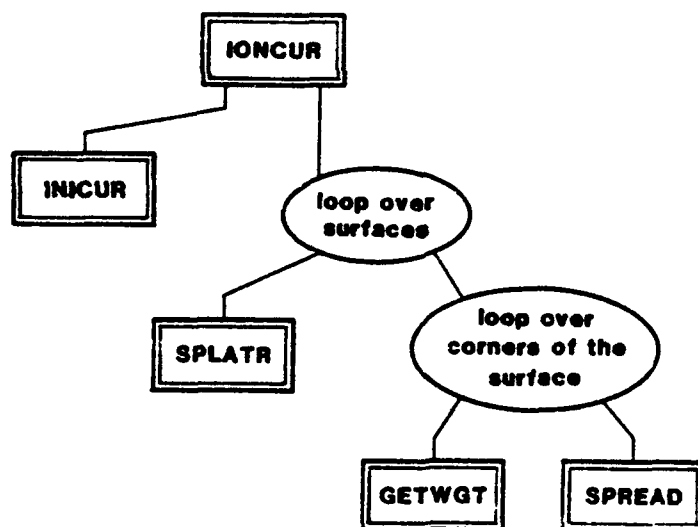


Figure 5.71/1. Structure diagram of IONCUR segment.

The creation of the SRFI list occurs in two main steps; step 1 reduces the dead-list to an intermediate SRFC list, and step 2 redistributes the surface currents among the surfaces. The justification and computational techniques for these steps have been discussed in Section 4.53.

Step 1 is controlled from IONCUR and performed by INICUR. The dead-list can be very large with an order that is best described as chronological. A particular surface will appear as often as it is struck by a particle. To speed up the surface current calculation, and to reduce the noise inherent to particle pushing, INICUR reads through the dead-list sequentially and simultaneously accumulates in the SRFC list the "average" particle (see 4.53) for each surface.

This average particle has the sum of the contributing current weights with an average impact location, energy, and velocity. These averages are weighted by the current weights (4.53).

Step 2 is the redistribution or sharing of the "raw" surface currents in the SRFC list. This is done to further reduce the "noise" in the raw ion surface currents (4.53). Thus, IONCUR will next loop over the surfaces in SRFC and for each surface, S, call SPLATR to calculate the bilinear weights, SPW (node), which are used to distribute the SRFC current to the vertices of the surface. These node currents are to be shared back to all of the surfaces adjoining each node. IONCUR loops over the surface vertices calling GETWGT to calculate the CRW (node, ns) where ns indexes neighboring surfaces (4.53). The surface-node-surface connectivity is provided by the LCEL list (5.25). Since the LCEL list is designed to be read sequentially and is ordered by slice and element, direct access to the needed section of LCEL is provided by an index list, SREL (5.25).

After the CRW have been obtained for a node, SPREAD is called to perform the summation;

$$\begin{aligned} \text{SRFI (ns)} &= \text{SRFI (ns)} \\ &+ \text{UNITS} * \text{CRW (node, ns)} * \text{SPW (NODE)} * \text{SRFC (S)} \end{aligned}$$

where ns ranges over the neighboring surfaces including surface S.

$$\text{UNITS} = \text{ECHRG} * \text{DXMESH}^2 * \text{FNORMI} * \sqrt{2\pi}, \text{ where } \text{ECHRG} = 1.6 \times 10^{-19}$$

Coulombs and DXMESH is the length of a mesh unit in meters. FNORMI is the unperturbed Maxwellian flux density against which the current weights are calibrated;

$$FNORMI = n_i \sqrt{kT/2\pi m_i}$$

where, in MKS, n_i is the ion density, k is Boltzmann's constant, T is temperature and m_i is the ion mass. For electrons, the equivalent to FNORMI, FNORME is used. For hydrogens, SRFH, SFHC, and FNORMH are used in place of SRFI, SRFC, and FNORMI, respectively.

5.72 CHARGE MODES

Three methods may be used to model the attracted species during charging calculations. Pushed particle surface currents, the default, may be used in space charge limited collection cases. Orbit limited collection currents may be used if appropriate.

An intermediate case is also available. This is the situation where the total current to the spacecraft is limited by a space charge limited sheath, but the actual surface currents are essentially orbit limited. The current through the sheath is calculated and saved. During the CHARGE cycle, the total orbit limited current to the object is computed. The two totals are then used to renormalize the orbit limited to each surface.

The selection of the appropriate charging mode is left to the expertise of the user. The default action is to assume space charge limited collection.

5.73 CHARGE (SURFACE CHARGING CONTROL)

CHARGE is the entry level routine into the surface charging section of POLAR (see Figure 5.73/1). It initializes the charge cycle by reading in the material properties and calculating the plasma to surface (routine MAKCSM) and surface to conductor (MAKCLG) capacitances as well as the conductance matrix (routine MAKSGM).

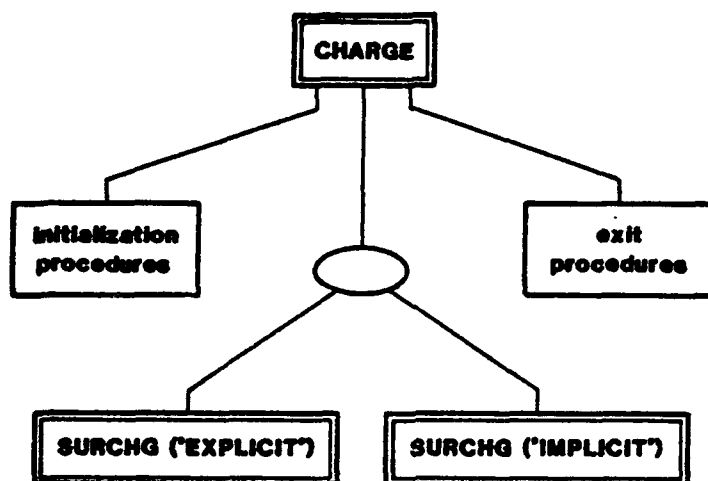


Figure 5.73/1. Structure diagram of CHARGE module.

After the initialization routines have been called, CHARGE calculates the surface potentials by finding a first stage solution then a second and final stage result. For historical reasons, the flags for the two stages are 'explicit' and 'implicit', respectively. CHARGE loops on the SURCHG sequence until converged or after a fixed number of iterations as defined by MAXITT (the loop is indicated by the empty oval). Upon conclusion, the array SURFV contains the new surface potential which is stored on file 19.

The computational theory for the charging section is discussed in Section 4.5.

5.73.1 SURCHG (SURFACE CHARGER)

SURCHG calls the routines needed to set up the equation to be solved and finds the next set of surface voltages. SURCHG solves the charging equation in the implicit form (Eq. (5.73-1)).

$$\left(\frac{C}{\Delta t} + g - \frac{dI}{dV} \right) \Delta V = I - g V \quad (5.73-1)$$

Using a flag sent by CHARGE, SURCHG determines which stage of the charging equation is being solved (see Figure 5.73/2). When solving the second stage charging equation (5.73-1), the SURCHG assumes a first stage solution had been found previously.

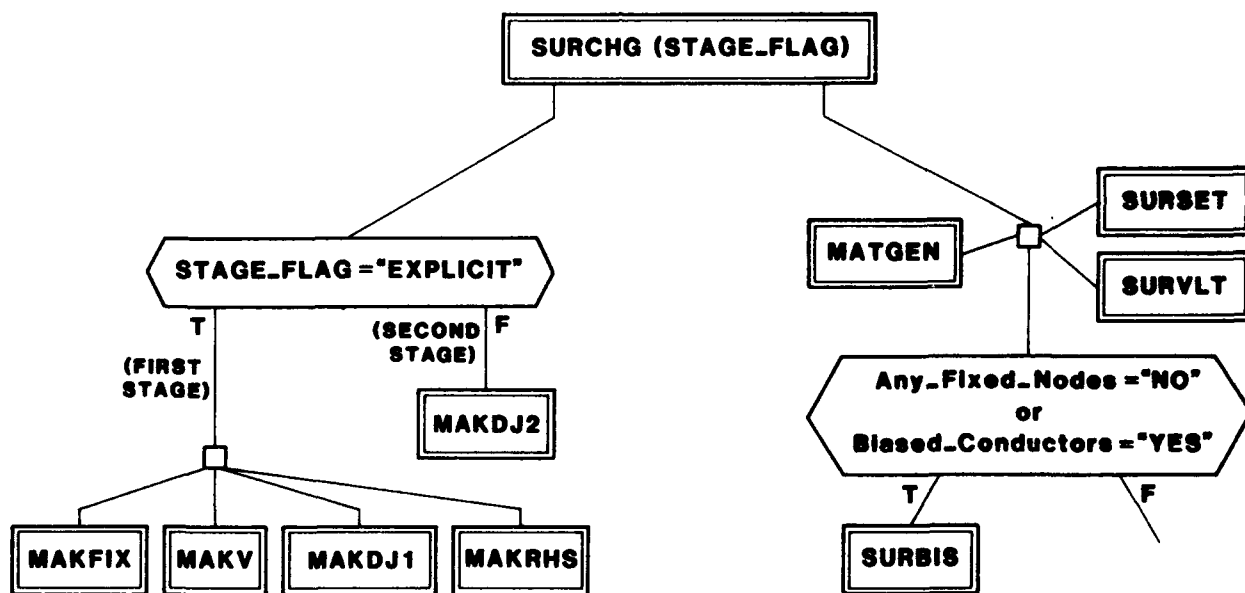


Figure 5.73/2. Structure diagram of the routine, SURCHG.

SURCHG, called with an "explicit" flag, generates the fixed surface vector (MAKFIX), V (MAKV), the first stage dI/dV (MAKDJ1), and $\tilde{I} - g \tilde{V}$ (MAKRHS), then calculates $V(t+\Delta t)$ for the first stage. When an implicit flag is received, MAKDJ2 is called to calculate the second stage $d\tilde{I}/d\tilde{V}$. MAKDJ2, after the first step in limiting the first stage voltage solution (Section 4.56.20), will call MAKJ2 which will calculate the currents to a surface at a given voltage. MAKJ2 will perform the second step of the voltage limiting process whenever it discovers a change in the sign of a surface's current from one stage to the next.

At this point, both charging equations can be treated the same way. The matrix,

$$\left(\frac{C}{\Delta t} + g - \frac{d\tilde{I}}{\tilde{V}} \right)$$

is calculated in the correct fixed or unfixed matrix form (Section 4.57) by MATGEN, setting up M for the ICCG call by SURVLT. SURVLT solves Eq. (5.73-2)

$$\tilde{M} \Delta \tilde{V} = \tilde{R} \quad (5.73.2)$$

where R is the vector found by MAKRHS and ΔV is the change in the surface potential.

Using the $\Delta \tilde{V}$ found by SURVLT, SURSET calculates the new surface potentials and saves them, completing the charging timestep. The following is a detailed discussion of the above routines.

MAKSGM (MAKE THE σ MATRIX)

MAKSGM makes the σ matrix by taking conductivity information created by VEHICL and moving it to the right place in the fixed matrix form (see 5.72.2) for ICCG after changing the units of the elements to mks units.

MAKFIX (MAKE THE FIXED SURFACE LIST)

MAKFIX constructs the fixed surface list. This list is used by ICCG to mark which nodes in the problem will not change, or change by fixed amounts. The routine MAKFIX also defines the voltage change, if any, of the fixed nodes.

MAKV (MAKE THE VOLTAGE VECTOR)

MAKV builds the voltage vector in the appropriate form (5.72.2) using a surface voltage list and the conductor voltage common block. The LAD number (Section 5.30) of the desired surface voltage list is passed as an argument to this routine.

MAKDJ1 (MAKE THE FIRST STAGE CURRENT DERIVATIVE)

MAKDJ1 uses the algorithm discussed in Section 4.56.20 to estimate the first stage current derivative.

MAKRHS (MAKE THE RIGHT HAND SIDE)

MAKRHS makes the right hand side of the vector equation solved by ICCG. The right hand side is equal to

$$R = I - g V .$$

This is done by first calling MAKJ to calculate the current vector. Then MAKRHS finds the product of σV and subtracts it from the current.

The magnetic field voltage effects and any conductor biasing terms (see Section 4.57) are calculated and subtracted as well.

The right hand side of the vector equation is the same for both stages so it only needs to be called once for each iteration.

MAKJ1 and MAKJ2 (MAKE THE CURRENT VECTORS)

These routines find the current vector I given a list of surface voltages. MAKJ1 is used during the first stage and MAKJ2 during the second. To find the contributions to the current from the various sources, MAKJ (MAKJ is the generic form of MAKJ1 and MAKJ2) calls the routines which calculate the incident, secondary, and backscattered electron fluxes and the incident and secondary ion currents.

Two lists are created by MAKJ, one is the total current, I_t . The other is the sum of the secondary currents, I_{sm} . The secondaries are used by MAKJ1 to locate surfaces which will need to be fixed.

MAKJ2 is called by MAKDJ2 (below). When this occurs, it checks for changes in the sign of the total current or more importantly, the proximity of the current equilibrium point. Then, using the algorithm described in Section 4.56.20, the surface voltage is further limited and a parabolic interpolation is done to find the crossover point. The current at this new voltage is defined to be zero for purposes of calculating second stage current derivatives.

MAKDJ2 (MAKE THE SECOND STAGE CURRENT DERIVATIVE)

MAKDJ2 is called during the second stage to provide the dI/dV matrix as described in Section 4.56.20. The second step of the voltage limiting process, performed prior to calculating the derivative, is done by MAKJ2 (above). MAKJ2 is also used to provide the currents at the limited surface potentials.

MATGEN (MATRIX GENERATION)

MATGEN finds the sum of the matrices which multiply the voltage vector, $\Delta \underline{V}$, solved for by ICCG. It performs the sum

$$\underline{\underline{M}} = \frac{\underline{\underline{C}}}{\Delta t} + \underline{\underline{g}} - \frac{d\underline{\underline{I}}}{d\underline{\underline{V}}}$$

If the desired matrix formulation is not the fixed form (Section 4.57), the σ matrix is manipulated to the proper form before the other matrices are added to it.

SURBIS (VECTOR REFORMULATOR, SURFACE BIAS)

SURBIS is invoked to change the vectors to the formulation appropriate to the situation (see Section 4.57). The matrix $\underline{\underline{M}}$ has already been reformulated by MATGEN though some additional changes are made when there is biasing in the problem.

SURVLT (SURFACE VOLTAGE)

SURVLT recovers from storage on file 19, the $\underline{\underline{M}}$ matrix and $\underline{\underline{R}}$ vector for use by ICCG. It also calls ICCG (4.32) which returns $\Delta \underline{\underline{V}}$. Currently, the initial guess for $\Delta \underline{\underline{V}}$ is $\Delta \underline{\underline{V}} = 0$.

SURSET (SURFACE VOLTAGE RESET)

SURSET uses the solution found by SURVLT to find the new surface and conductor voltages after charging.

5.73.2 CHARGING MATRIX AND VECTOR FORMULATION

As discussed in Section 4.57, there are two cases which affect the charging equation, 5.73-1, depending on the occurrence or non-occurrence of fixed potential surfaces. When a surface potential is fixed, the matrices and vectors of 5.73-2 are constructed exactly as implied. If a surface has been fixed, (called from SURCHG) SURCHG passes a flag, ANYFIX, to the matrix vector routine, MATGEN, to construct the diagonalized matrices, \tilde{C} and \tilde{g} . These transformations are discussed in 4.5, but they are characterized by constructing $V(t)$ from the potential difference between a surface and the underlying conductor instead of simply the surface potential. The effect of this is to reduce the off-diagonal elements and increase the diagonal elements of the amalgamated matrix \tilde{M} (Eq. (5.73-2)). ICCG likes this and will generally converge faster.

Since there can be as many as 1200 surfaces or so, it was necessary to pack all the matrices in the problem (unpacked it could be 1 million words/matrix). Since the matrices are mostly empty, just saving the nonzero entries and their locations simplifies the problem greatly. The entry locations are stored in a list (called 'LIST' in MRBUF) with a second list ('MRKR') being used to indicate the starting locations of the rows in LIST. The entry locations are stored row by row with the first entry for a row being the negative row number. For example, the nonzero locations in the following matrix

$$\begin{pmatrix} a & 0 & 0 & 0 & 0 \\ 0 & a & b & 0 & 0 \\ 0 & b & a & e & f \\ 0 & c & e & a & d \\ 0 & 0 & f & d & a \end{pmatrix}$$

would appear in LIST as -1, -2, 3, -3, 2, 4, 5, -4, 2, 3, 5, -5, 3, 4 and MRKR would be 1, 2, 4, 8, 12, 10000 with the 10000 signalling the end. Since the matrices always have nonzero diagonals, the -2 in LIST means there is an entry at (2, 2). The 3 following it means (2,3) and so on. The actual matrix values are stored in lists so that the locations of the entry in the LIST correspond to the location of the value. For example, the value b stored at (2, 3) would be the third value in the value list. For large problems this greatly reduces the data requirements.

5.80 OUTPUT

5.81 GENERAL

Output is produced throughout the POLAR modules. The various print statements placed in the code during development were flagged rather than removed when development was completed. This allows output to generate at many levels of verbosity. These "diag" flags are discussed in detail in Sections 6.22, 6.31, 6.44, and 6.45.10. Utilities have also been created which print only selected Z slices of the grid data (see Section 6.44, the "SELECT" keyword).

During VEHICL and NTERAK runs, the two modules write notes to a file called STATUS.JCO. The notes are date and time stamped and contain short informative facts. The STATUS.JCO file is opened, added to, and then closed. Because it is always up to date, it provides a quick, concise means to follow the progress of a calculation. The file also summarizes the run history of a POLAR calculation.

Postprocessing diagnostics and graphical output is available from the SHONTL program. The "PRINT" keyword produces data stored on the buffered I/O files 11 and 19. SHONTL also generates color or black and white pictures of space potentials and several types of space charge densities.

TRMTLK can be used to postprocess the charging calculation data on file 16.

5.82 GRAPHICAL CODE STRUCTURE

5.82.10 AN OVERVIEW OF SHONTL

SHONTL is a module designed to create the graphics commands for PLOTTR. It presents cross-sections of data generated by NTERAK as contour plots. Currently, axial slices of the ion density, the electric field potential, the sheath boundary, and the particle trajectories can be plotted (DION, DELC, GI, GH, QUSD, and POT).

The form of the plots can be varied by using keyword commands. The plot types and slice locations are also defined by keywords. The keywords and their use are described in Section 6.5.

5.82.11 SHONTL

This is the main routine of the module. SHONTL oversees the order of operations of the various phases of the plotting process (see Figure 5.82/1). It calls an initialization routine (SHODEF), an exit routine (SHOXIT) and cycles on the input routine and the plotting routine (SHOINP and GENPLT, respectively). EOF is a flag from SHOINP signalling the end of a plotting session.

5.82.12 SHODEF (PLOT INITIALIZATION)

SHODEF is the initialization routine. It opens POLAR data files 11 and 19, sets the faster graphics window, initializes common blocks, sets the defaults for plot descriptions and prints a welcoming message.

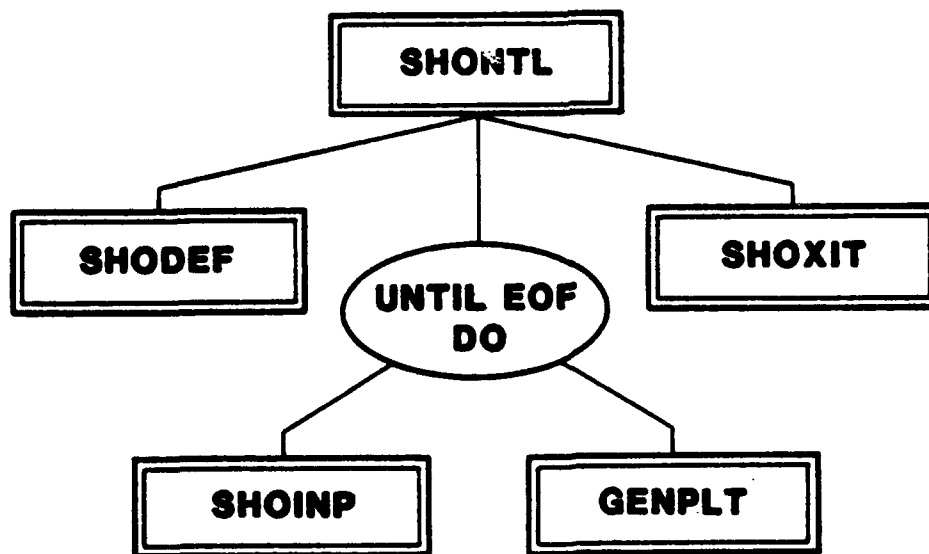


Figure 5.82/1. A general structure diagram of SHONTL.

5.82.13 SHOINP (SHONTL INPUT)

This routine is the keyword input section of SHONTL. SHOINP recognizes the various keywords (described in detail in Section 6.5) and sets the appropriate flag for GENPLT. In addition to keywords for plotting features and to describe the data for plotting, keywords are also recognized which direct the data handling, control the SHONTL exiting and plotting procedure, and provide useful aids for the user. This includes a 'WHAT' command to see which plotting features have been selected.

All of the plotting features have default values set in SHOINP so that no harm will be done if a flag is not set in SHOINP (the default values are described in Section 6.5 also).

5.82.14 GENPLT (GENERATE PLOTS)

GENPLT calls the routines responsible for generating the pseudo-graphics calls used by PLOTTR. The flags set by SHODEF and SHOINP are used to decide which plotting routines to call.

To make a plot, GENPLT sets the constants it will need, reads the data to be plotted, generates the plot along with any extra features selected, and then prepares for the next plot.

5.82.15 SHOXIT (SHONTL EXIT)

SHOXIT is the exit routine for SHONTL.

REFERENCE, CHAPTER 5

- 5-1 "NASCAP Programmers' Reference Manual," S-CUBED Report SSS-R-82-5443 (DRAFT), March 1982.

6.0 OPERATING INSTRUCTIONS

In general, POLAR can be run in either an interactive or batch computing environment using keyword input. Keyword values are set to defaults during object definition (VEHICL) and are passed from module to module and run to run. New input is remembered, so it is not necessary to re-enter the same keywords constantly. Commands controlling run characteristics, defining environmental constants, and choosing the amount of appropriate output are either recognized by the module's input routine or by a general input routine (POLINP) which processes non-module specific keywords such as the grid size and diagnostic keywords.

The exception to this general rule is the set of keywords used to define an object. These keywords are expected to be found within a definition file separate from the VEHICL runstream. Section 6.1 describes the intricacies of object definition in detail and includes several helpful examples.

The other sections describe the keywords pertinent to all of the modules (Section 6.7) or keywords appropriate to a particular module (Sections 6.2 to 6.5). Some keywords appear in several locations for the sake of convenience.

6.10 OBJECTS

POLAR objects are defined separately from the VEHICL runstream. VEHICL will look for the object definition file as unit 20 (Fort.20).

POLAR objects are defined in an object grid of variable proportions subject to the limitation $NX \cdot NY \cdot NZ = 9537$; $NY, NX \leq 17$; $NZ \leq 33$. The keyin language is identical to NASCAP, except for the addition of slanted thin plates and the omission of booms (POLAR does not do booms), antennas, and thin triangular plates. If "empty space" and "object" both coexist in the same computational space, what makes objects distinguishable? The answer is that POLAR can distinguish between volume elements that are filled (with object) and those that are empty (except for ambient plasma of course). Once we have this distinction it is easy to see how objects can be constructed by filling in collections of volume elements. For example, a simple cuboid may be constructed by filling in $2 \times 3 \times 4 = 24$ elements as shown in Figure 6/1.

While arrangements of completely filled and completely empty cubes can be quite versatile in representing objects of many different shapes, more sophisticated representations are possible if we allow cubes to be partially filled (or partially empty). Only three partially filled cubes are allowed. These are shown in Figure 6/2.

While it is easy to see how objects might be constructed by filling or partially filling individual volume elements, a command structure that required the user to specify every element comprising an object would be very cumbersome to use. So several generalized objects are definable.

It is very important to note that POLAR objects must never touch the edge of the object grid defined by VEHICL (see Section 6.2).

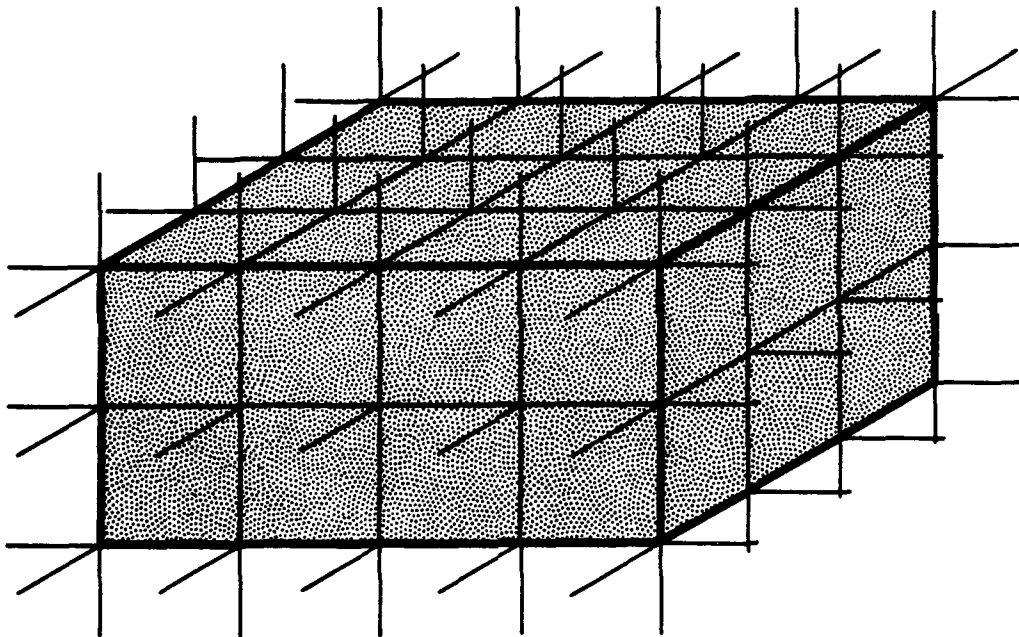
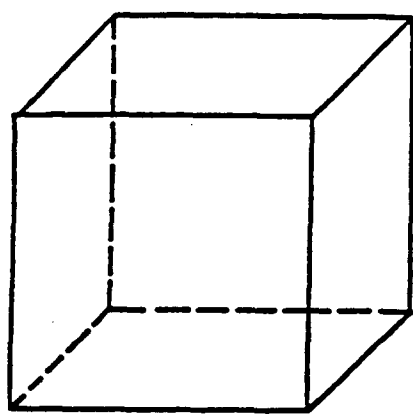
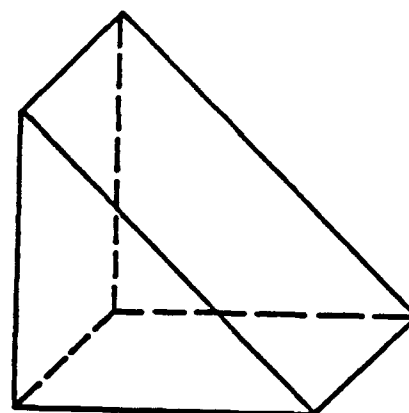


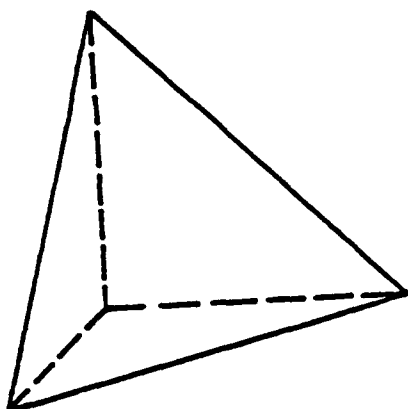
Figure 6/1. Cuboid made by filling in twenty-four volume elements.



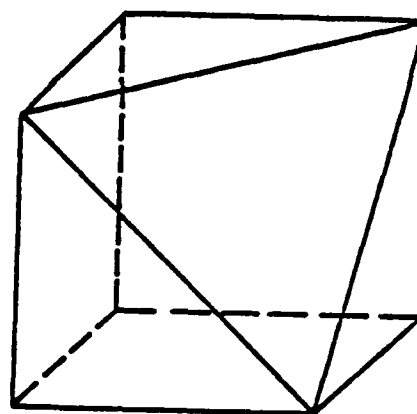
a



b



c



d

Figure 6/2. Four shapes of volume cells considered by the POLAR CODE:
(a) empty cube; (b) wedge-shaped cell with 110 surface;
(c) tetrahedron with 111 surface; (d) truncated cube with 111 surface.

6.10.10 BUILDING BLOCKS

To greatly simplify the user definition of objects, POLAR pre-defines commonly used shapes built up from individual elements. These shapes are called POLAR BUILDING BLOCKS. There are eight.

Flat Plate

Slanted Plate

Cuboid

Octagon

Quasisphere

Tetrahedron

Wedge

FIL111

These are shown in Figure 6/3. These basic shapes can be defined to be any size (within the inner grid). POLAR automatically includes the correct number of individual elements for the size of building block chosen by the user.

6.10.11 COMMANDS (OR HOW DO I ACTUALLY DEFINE AN OBJECT?)

The POLAR module VEHICL is responsible for recognizing and understanding the object defined by the user in the object definition file.

Each building block has its own keyword. For example, the quasi-sphere is associated with the word QSPHERE, and the cuboid (rectangular parallelepiped) with the word RECTAN. The building blocks and their keywords are summarized in Table 6/1.

Once VEHICL has read a building block keyword from the object definition file, it then expects to find several more lines (or cards) setting the block parameters. These might include the dimensions of

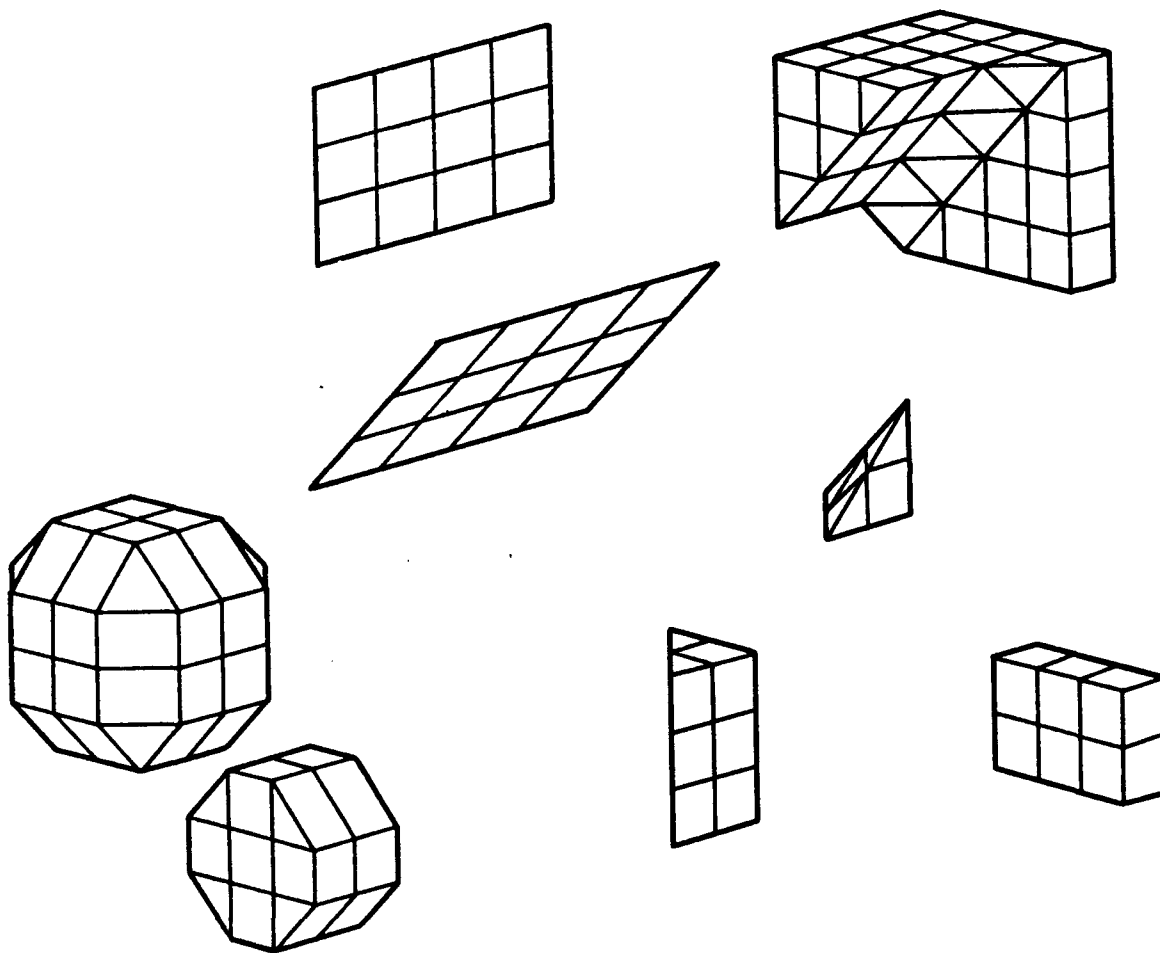


Figure 6/3. The eight building block types are shown here. The uppermost object shows a FIL111 smoothing a corner. Below, from left to right are quasi-sphere, octagon right cylinder, tetrahedron, wedge, and rectangular parallelepiped.

TABLE 6/1. POLAR BUILDING BLOCKS AND THEIR KEYWORDS

<u>Keyword</u>	<u>Building Block Description</u>
FIL111	Smooth inside of a diagonal corner
OCTAGON	Right octagonal cylinder
PATCHR	Surface of a rectangle
PATCHW	Diagonal face of a wedge
PLATE	Arbitrarily thin plate or cuboid
QSPHERE	Quasisphere
RECTAN	Cuboid or rectangular parallelepiped
SLANT	Thin plate slanted at 45
TETRAH	Tetrahedron
WEDGE	Wedge derived from half a cube

the building block, its orientation and the materials that cover its surface. (Surface materials are discussed in Section 6.12.) Finally, VEHICL expects to find a line 'ENDOBJ' telling it that no more information referring to the present block is coming and to expect the next building block keyword. The information to be entered in the object definition file for each building block is summarized in Table 6/2. Note that numbers and words may be separated by one or more spaces on the same line. (Input is free-format.)

The keyword 'ENDSAT' signals the end of the satellite definition and should be included at the end of all vehicle descriptions.

6.10.12 PLATES AND PATCHES

A careful inspection of Table 6/1 will show that there are some building blocks that are not derived from cubic volume elements. These are the PLATE, SLANT, PATCHR and PATCHW.

PLATEs are arbitrarily thin cuboids (RECTANS). They are assumed to have only a top and a bottom, the sides being of negligible height. Flat plates always lie in one of the axis planes (XY, XZ, YZ). SLANTED plates lie along one axis, and at a 45° angle to the other two.

PATCHR and PATCHW are the surfaces only of a cuboid and wedge, respectively. They are used to change the surface material patterns of existing building blocks and should never be defined in spaces not already occupied by solid objects. (Objects defined to occupy the same space are explained in Section 6.14.)

TABLE 6/2. OBJECT DEFINITION - FILE 20

OBJECT DEFINITION
SYNTAX

RECTAN
CORNER x y z
DELTAS $\Delta x \Delta y \Delta z$
(UP TO 6 SURFACE CARDS)
ENDOBJ

WEDGE
CORNER x y z
FACE materialname normal
(type 110)
LENGTH $\Delta x \Delta y \Delta z$
(UP TO 4 SURFACE CARDS)
ENDOBJ

TETRAH
CORNER x y z
FACE materialname normal
(type 111)
LENGTH Δx
(UP TO 3 SURFACE CARDS)
ENDOBJ

OCTAGON
AXIS x y z x' y' z'
WIDTH w
SIDE s
(UP TO 3 SPECIAL SURFACE
CARDS "+" "-" or "C")
ENDOBJ

OSPHERE
CENTER x y z
DIAMETER d
SIDE s
MATERIAL materialname
ENDOBJ

SLANT
CORNER x y z
TOP materialnormal
(type 110)
BOTTOM material
LENGTH $\Delta x \Delta y \Delta z$
ENDOBJ

OBJECT DEFINITION
EXAMPLES

RECTAN
CORNER 3 -2 8
DELTAS 1 2 4
SURFACE +X ALUMINUM
SURFACE -X ALUMINUM
SURFACE +Y ALUMINUM
SURFACE -Y ALUMINUM
SURFACE +Z ALUMINUM
SURFACE -Z ALUMINUM
ENDOBJ

WEDGE
CORNER -3 2 1
FACE SIO2 -1 -1 0
LENGTH 1 1 3
SURFACE +X SIO2
SURFACE +Y SIO2
SURFACE +Z GOLD
SURFACE -Z SIO2
ENDOBJ

TETRAH
CORNER -3 -2 8
FACE KAPTON 1 1 -1
LENGTH 1 2
SURFACE -X TEFLON
SURFACE -Y KAPTON
SURFACE +Z TEFLON
ENDOBJ

OCTAGON
AXIS 3.2 -6 3.2 -9
WIDTH 3
SIDE 1
SURFACE + SILVER
SURFACE - SILVER
SURFACE C MAGNES
ENDOBJ

OSPHERE
CENTER 0.0.0
DIAMETER 4
SIDE 2
MATERIAL NPAIN
ENDOBJ

SLANT
CORNER -1 -1 0
TOP TEFLON 1 1 0

BOTTOM KAPTON
LENGTH 2 2 3
ENDOBJ

OBJECT DEFINITION
SYNTAX

FIL111
CORNERLINE x y z x' y' z'
FACE materialname normal
(type 111)
ENDOBJ

PLATE
CORNER x y z
DELTAS $\Delta x \Delta y \Delta z$
TOP $\pm \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ materialname
BOTTOM $\pm \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ materialname
ENDOBJ

PATCHR
CORNER x y z
DELTAS $\Delta x \Delta y \Delta z$
(UP TO 6 SURFACE CARDS)
ENDOBJ

PATCHW
CORNER x y z
FACE materialname normal
(type 110)
LENGTH $\Delta x \Delta y \Delta z$
(UP TO 4 SURFACE CARDS)
ENDOBJ

OBJECT DEFINITION
EXAMPLES

FIL111
CORNER 3.2.6 -5 4.6
FACE SOLAR -1 -1 -1
ENDOBJ

PLATE
CORNER -1 -1 -10
DELTAS 2 2 0
TOP +Z CPAINT
BOTTOM -Z CPAINT
ENDOBJ

PATCHR
CORNER 3 -2 8
DELTAS 1 0 1
SURFACE -Y SCREEN
ENDOBJ

PATCHW
CORNER -3 2 7
FACE AQUADG -1 -1 0
LENGTH 1 1 1
ENDOBJ

NOTES "normal" is three values.
each either +1, 0, or -1.

SURFACE CARD has the following format:

SURFACE $\pm \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ materialname

SPECIAL SURFACE CARD is:

SURFACE $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ materialname

OTHER OBJECT DEFINITION COMMANDS

ENDSAT	Must be last card in file
COMMENT	No effect.
OFFSET i j k	Moves coordinate origin
CONDUCTOR n	Sets number of underlying conductor ($1 \leq n \leq 15$).
DELETE i j k	Deletes surfaces, leaving empty cell
unrecognized word	Assumed to be name of new surface material. Next card scanned for parameters.

6.10.13 SPECIAL SHAPES

FIL111 is a special shape designed to fill in "steps" whose corner line runs at 45° to the grid lines in any axis plane (i.e., XZ, ZY, XZ) (Figure 6/4a). There are two kind of "steps" that can occur between POLAR building blocks. For example, a small cuboid on top of another creates four "steps" that lie along grid lines (Figure 6/4b). These may be "filled in" or smoothed by defining a WEDGE to lie along the corner line of the step. A second type of step is possible however when, for example, a tetrahedron or octagon is defined to sit on top of another building block. These steps have corner lines that run at 45° between grid lines. This is shown in Figure 6/4c. Such steps can be smoothed or filled in by a combination of tetrahedra and truncated cubes. This combination is supplied as the building block FIL111.

6.10.14 BUILDING BLOCK PARAMETERS (OR WHO'S ON NEXT?)

The very center of the object grid is assumed to be the origin of the coordinate system 0, 0, 0. Hence the grid itself extends from -8 to +8 in the X and Y directions and from -16 to +16 in the Z direction. This coordinate system is used to specify the position and size of the building blocks in the parameter "cards" or lines following the building block keyword. Let us examine the definition of each building block in detail to see how this works.

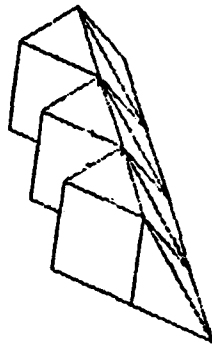


Figure 6/4a. A FIL111 building block all by itself.

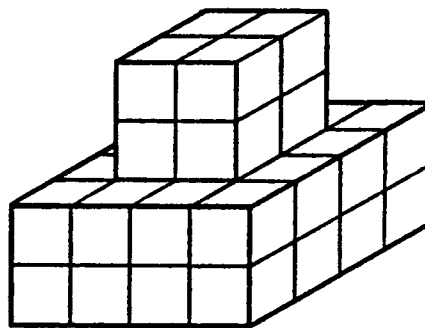


Figure 6/4b. "Steps" along grid lines.

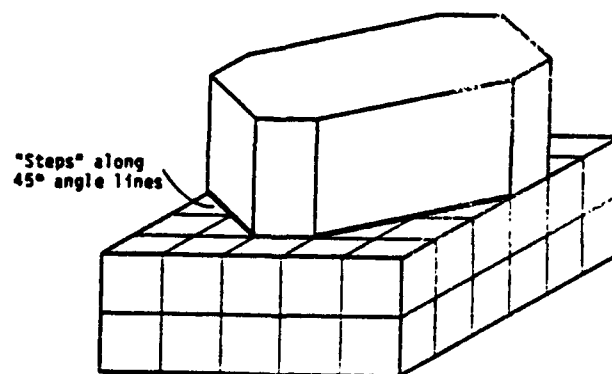


Figure 6/4c. "Steps" along 45° angle lines.

6.10.15 RECTAN

The following cards define a cuboid or rectangular parallelepiped:

RECTAN

CORNER x y z

DELTAS Δx , Δy , Δz

SURFACE +X GOLD

SURFACE -Y KAPTON

(Four more SURFACE cards for -X, +Y, +Z, -Z)

ENDOBJ

Notes:

1. RECTAN: is the building block keyword.
2. CORNER x y z: defines the coordinate of the lowest indexed corner of the cuboid (the one so that if you added up $x + y + z$ it would give the lowest (least positive) number).
3. DELTAS Δx , Δy , Δz : gives the length of sides of the cuboid along the X, Y and Z axes. (Note that the edges of the cuboid must lie in the direction of the three axes.)
4. SURFACE +X GOLD: assigns the material GOLD (see Chapter 6.12) to the surface of the cuboid whose normal points in the +X direction. There are up to six surfaces that may be assigned materials (+X, -X, +Y, -Y, +Z, -Z). All surfaces that will eventually become a surface of the finished object (rather than become a connection to another building block) must be assigned a material. (For surfaces that are shared with other building blocks the material assigned is ignored.)

As an example, the following cards:

```
RECTAN
CORNER  -4  2  -1
DELTAS   3  2   5
SURFACE  +X  GOLD
SURFACE  +Y  GOLD
SURFACE  +Z  GOLD
SURFACE  -X  GOLD
SURFACE  -Y  GOLD
SURFACE  -Z  GOLD
ENDOBJ
```

define a gold bar extending from -4 to -1 in the X direction, 2 to 4 in the Y direction and -1 to +4 in the Z direction (Figure 6/5).

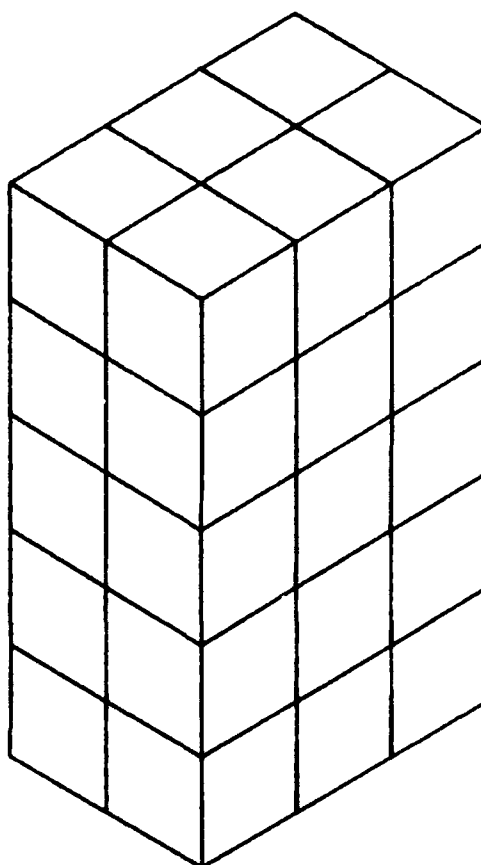


Figure 6.5. RECTAN.

6.10.16 PATCHR

PATCHR is defined in exactly the same way as RECTAN.

```
PATCHR
CORNER  x y z
DELTAS  Δx Δy Δz
<SURFACE card(s) (usually just one)j
[e.g., SURFACE +X GOLD
ENDOBJ
```

PATCHR should only be defined within an existing object. (see 6.14).

6.10.17 WEDGE

The following cards define a right angled wedge:

```
WEDGE
CORNER  x y z
FACE    KAPTON 1 1 0
LENGTH  Δx Δy Δz
SURFACE +X TEFLON
<Up to four SURFACE cardsj
ENDOBJ
```

Notes:

1. WEDGE: is the building block keyword.
2. CORNER x y z: defines the lowest indexed vertex of the right angled corner of the wedge (see note 2, 6.10.15).
3. FACE KAPTON 110: contains two pieces of information:
 - a. 'KAPTON' assigns the material KAPTON to the surface of the face of the wedge. (The face is the sloping surface of the wedge.)
 - b. '1 1 0' defines the direction of the normal to the face and hence the orientation of the wedge itself. The normal may point in any of the following directions only:

$\underline{+1}, \underline{+1}, 0$

$\underline{+1}, 0, \underline{+1}$

$0, \underline{+1}, \underline{+1}$

(For those of you not familiar with the '1 1 0' notation a '1 1 0' normal is a vector pointing to the coordinates $X = 1$, $Y = 1$ and $Z = 0$ from the origin.)

4. LENGTH $\Delta x, \Delta y, \Delta z$: gives the lengths of the sides of the wedge parallel to the X, Y and Z axes. To maintain symmetry two of these must be equal (i.e., the two right triangle sides).
5. SURFACE +X TEFLON: assigns the material 'TEFLON' (Section 6.12) to the surface whose normal points in the positive X direction. There are up to four remaining surfaces that may be assigned materials (see 6.10.15, note 4). These all have normals pointing along one of the axis directions. Along which axis direction they point depends on the orientation of the wedge or the choice of normal for the face (note 2). The possible combinations of face directions and remaining surface directions are summarized in Table 6/3. Cards defining materials for non-existent faces are ignored.

As an example, the following cards:

```

WEDGE
CORNER  0 0 0
FACE    GOLD  1 1 0
LENGTH  2 2 2
SURFACE -X  GOLD
SURFACE -Y  GOLD
SURFACE +Z  GOLD
SURFACE -Z  GOLD
ENDOBJ

```

define a wedge covered in gold with the origin as one of its corners and a face whose normal points between the X and Y axes in the XY plane. This is shown in Figure 6/6.

TABLE 6/3. DIRECTIONS OF SURFACE NORMALS ASSOCIATED WITH
ALLOWED WEDGE ORIENTATION

<u>Normal of WEDGE Face</u>	<u>Normals of Four Remaining Surfaces</u>
1 1 0	-X, -Y, Z, -Z
-1 1 0	X, -Y, Z, -Z
1 -1 0	-X, Y, Z, -Z
-1 -1 0	X, Y, Z, -Z
1 0 1	-X, Y, -Y, -Z
-1 0 1	X, Y, -Y, -Z
1 0 -1	-X, Y, -Y, Z
-1 0 -1	X, Y, -Y, Z
0 1 1	X, -X, -Y, -Z
0 -1 1	X, -X, Y, -Z
0 1 -1	X, -X, -Y, Z
0 -1 -1	X, -X, -Y, -Z

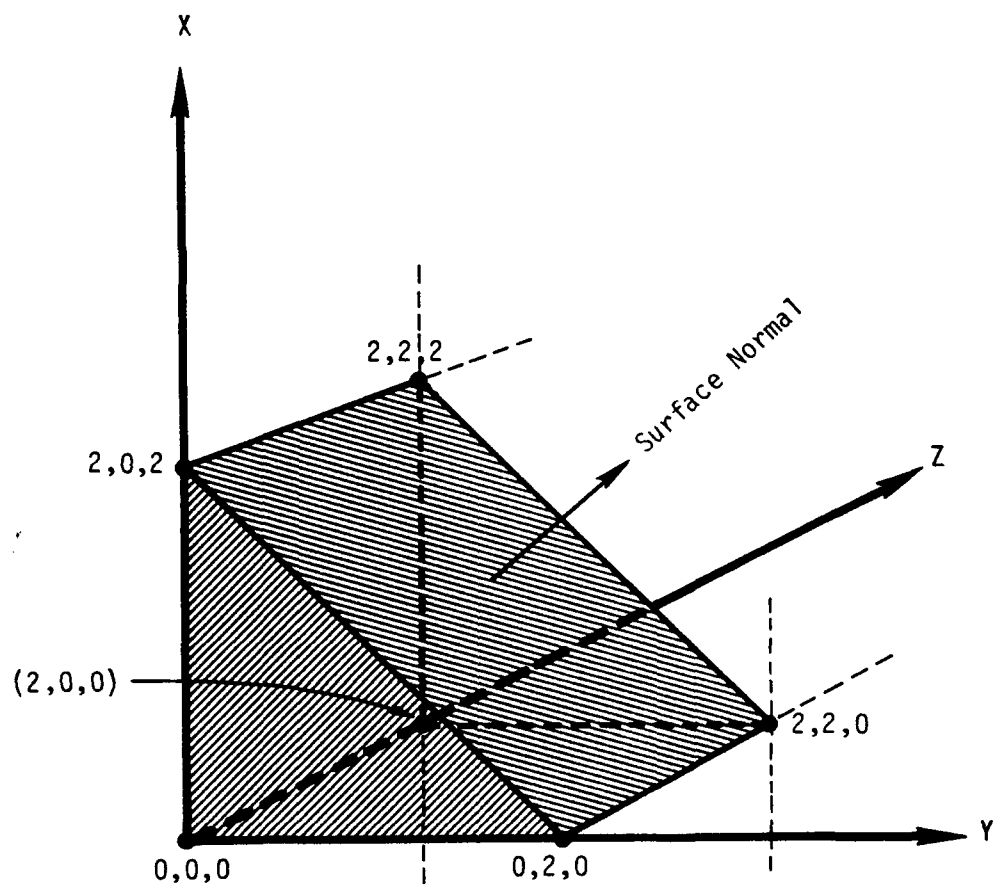


Figure 6/6. Wedge defined with surface normal 110 and corner 0,0,0.

6.10.18 PATCHW

PATCHW is defined in exactly the same way as a wedge.

```
PATCHW
CORNER  x y z
FACE  GOLD  1  -1  0
LENGTH  Δx  Δy  Δz
<Up to four SURFACE cards (usually just one)>
ENDOBJ
```

Like PATCHR (6.10.16) it may only be used to define a wedge inside another building block. This is explained further in Section 6.14.

6.10.19 TETRAH

The following cards define a tetrahedron:

```
TETRAH
CORNER  x y z
FACE  ALUMINUM  1  1  -1
LENGTH  Δx
SURFACE  -X  TEFLON
SURFACE  -Y  TEFLON
SURFACE  +Z  TEFLON
ENDOBJ
```

Notes:

1. TETRAH is the building block keyword.
2. CORNER x y z: defines the coordinates of the right angled corner of the tetrahedron. There is only one of these. (It corresponds to the corner of the partially filled cubic volume element that is actually filled.)
3. FACE 1 1 -1: assigns the material ALUMINUM to the unique face of the tetrahedron opposite the right angled corner. '1 1 -1' gives the direction of this face's surface normal and hence the orientation of the tetrahedron. The following directions only are allowed:

+1, +1, +1

(This notation is the same as explained in 6.10.17, note 3.)

4. LENGTH Δx : gives the length of the sides along the X, Y and Z axis directions. (These must all be equal to preserve symmetry.)
5. SURFACE -X TEFLON: assigns the material teflon to the remaining surface with surface normal pointing along the negative X axis direction. Up to three surfaces remain to be assigned materials (see 6.10.15, note 2). The surface normals of these surfaces depend on the orientation of the tetrahedron and hence the normal of the "face". Table 6/4 summarizes these relationships. Definitions of non-existent surfaces are ignored.

TABLE 6/4. DIRECTIONS OF SURFACE NORMALS ASSOCIATED WITH
ALLOWED TETRAHEDRON ORIENTATIONS

<u>Normal of TETRAHedron Face</u>	<u>Normals of Three Remaining Surfaces</u>
1 1 1	-X, -Y, -Z
-1 1 1	X, -Y, -Z
1 -1 1	-X, Y, -Z
1 1 -1	-X, -Y, Z
-1 -1 1	X, Y, -Z
-1 1 -1	X, -Y, Z
1 -1 -1	-X, Y, Z
-1 -1 -1	X, Y, Z

As an example, the following cards:

```
TETRAH
CORNER 0 0 0
FACE KAPTON 1 1 1
LENGTH 2
SURFACE -X KAPTON
SURFACE -Y KAPTON
SURFACE -Z KAPTON
ENDOBJ
```

define a tetrahedron with its right angle corner at the origin and the normal of the opposite face pointing between the positive X, Y and Z axes. This is shown in Figure 6/7.

6.10.20 OCTAGON

The following cards define a right octagonal cylinder:

```
OCTAGON
AXIS x y z x' y' z'
WIDTH w
SIDE s
SURFACE + GOLD
SURFACE - GOLD
SURFACE C GOLD
ENDOBJ
```

Notes:

1. OCTAGON: is the building block keyword.
2. AXIS x y z x' y' z': defines both the direction of the symmetry axis and the height of the cylinder. The symmetry axis must be parallel to one of the axis directions. Thus two of the coordinate pairs (x, x'), (y, y') and (z, z') must be identical. For example,

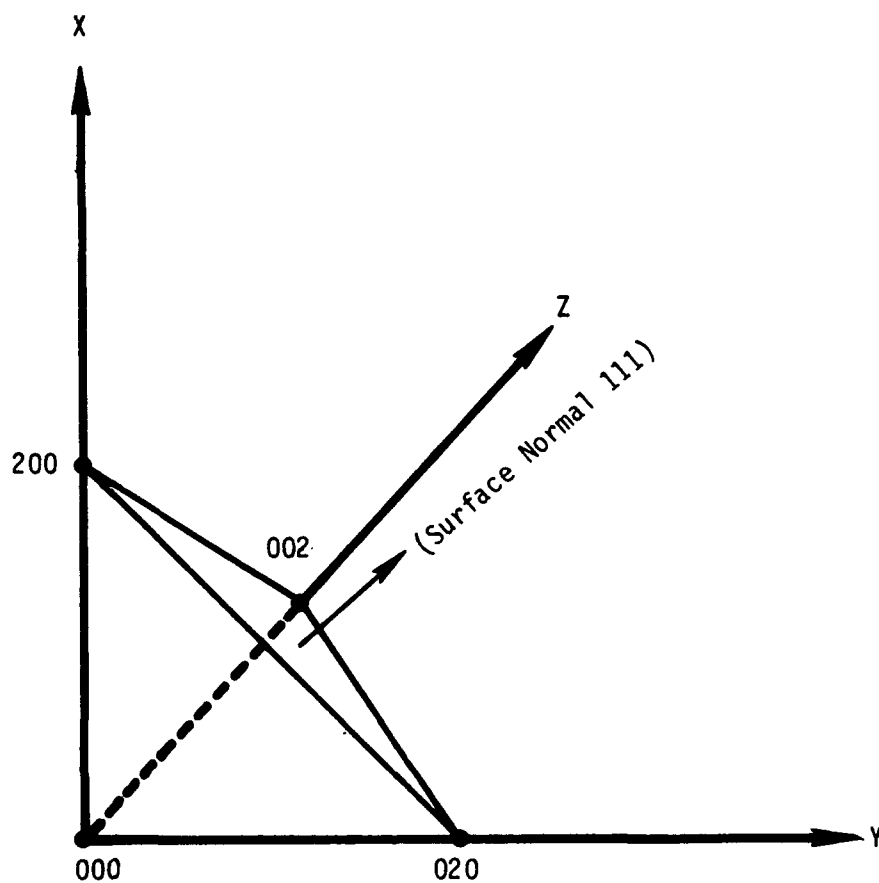


Figure 6/7. Tetrahedron defined with its "corner" at 000 and a surface normal 111.

'AXIS 6 3 -2 7 4 -2'

would define an axis that was not parallel to the X, Y or Z directions. However,

'AXIS 6 3 -2 7 3 -2'

defines an axis parallel to the X direction and is acceptable.

The height of the cylinder is given by the difference in coordinates along the axis direction. (For example, in the case above, the axis is one mesh unit long.)

3. WIDTH w: gives the width of the octagonal cross-section of the cylinder as w. If WIDTH is chosen to be odd, the axis must be moved or the sides of the cylinder will lie halfway across a volume element. POLAR automatically moves the axis $+1/2$ a mesh unit in each direction in the plane perpendicular to it.
4. SIDE s: gives the length of one of the sides of the octagonal cross-section that lies in an axis direction. The symmetry relationship between the width and the sides of the cross-section is shown in Figure 6/8. To maintain this relationship the side must always be an even number of mesh units less than the width. This means that they both either must be odd or both even numbers of mesh units.
5. SURFACE + GOLD: assigns the material GOLD to the top surface of the cylinder. '-' and 'C' replacing the '+' assign surface materials to the bottom or side cylindrical surface, respectively. Only those surfaces that will eventually become surfaces of the completed object need be assigned a material.

As an example, the following cards:

```
OCTAGON
AXIS 2 -4 6 2 -4 10
WIDTH 5
SIDE 3
SURFACE + TEFLON
SURFACE - TEFLON
SURFACE C TEFLON
ENDOBJ
```


defines a right octagonal cylinder covered in teflon. The symmetry axis lies along the Z direction and the height of the cylinder is four mesh units. Because the WIDTH is odd the axis is imagined to pass through the point $2\frac{1}{2}$, $-3\frac{1}{2}$ in the X Y plane. Hence the top and bottom faces run from $X = 0$ to $X = 5$ and from $Y = -6$ to $Y = -1$. The coordinates of the top of the cylinder are shown in Figure 6/8. A three-dimensional view is shown in Figure 6.9.

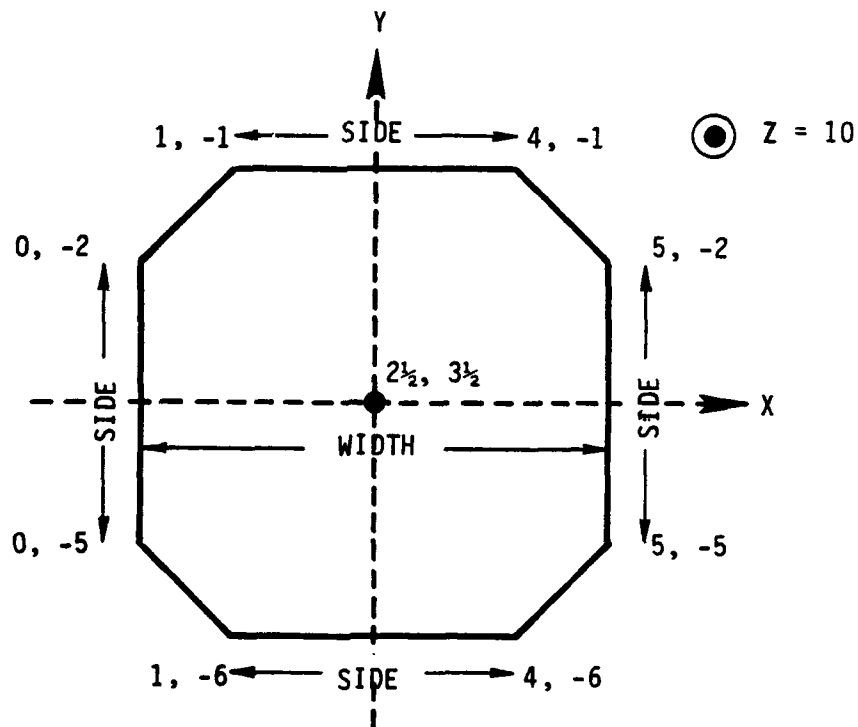


Figure 6/8. Top of an OCTAGON.

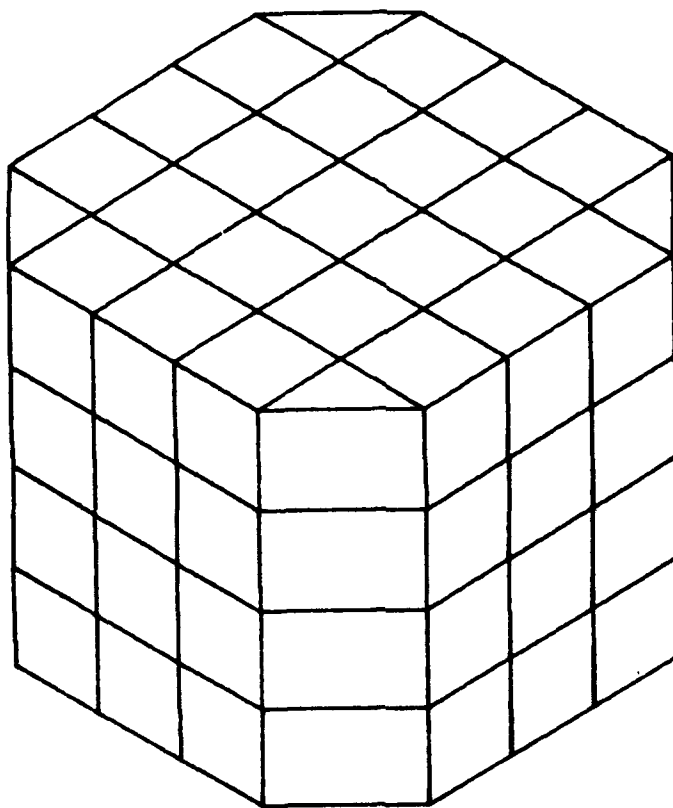


Figure 6/9. Octagon.

6.10.21 QSPHERE

The following cards define a quasisphere:

```
QSPHERE
CENTER x y z
DIAMETER d
SIDE s
MATERIAL SiO2
ENDOBJ
```

Notes:

1. QSPHERE: is the building block keyword.
2. CENTER x y z: defines the center of the sphere to be at coordinates X, Y, Z.
3. DIAMETER d: defines the diameter of the sphere to be d mesh units. The quasisphere can be thought of as an octagonal cross-section (like the top of an OCTAGON (see 6.10.20)) rotated about an axis in the cross-section plane. The diameter then corresponds to the WIDTH for a two-dimensional octagonal section. The same restrictions then apply: An odd value for the DIAMETER causes POLAR to automatically move the CENTER by +1/2 a mesh unit in the X, Y and Z directions.
4. SIDE s: sets the length of a side lying in one of the axis planes (e.g., X Y plane). Like the OCTAGON, the SIDE and DIAMETER must differ by an even number of mesh units.
5. MATERIAL SiO2: assigns the material SiO2 to the whole sphere surface.

As an example, the following cards:

```
QSPHERE
CENTER 1 -3 5
DIAMETER 7
SIDE 3
MATERIAL SILVER
ENDOBJ
```

define a silver sphere centered at $1 \frac{1}{2}$, $-2 \frac{1}{2}$ and $5 \frac{1}{2}$. The sphere extends along the axis direction as follows:

x from -2 to 5

y from -6 to 1

z from 2 to 9

(See Figure 6/10.)

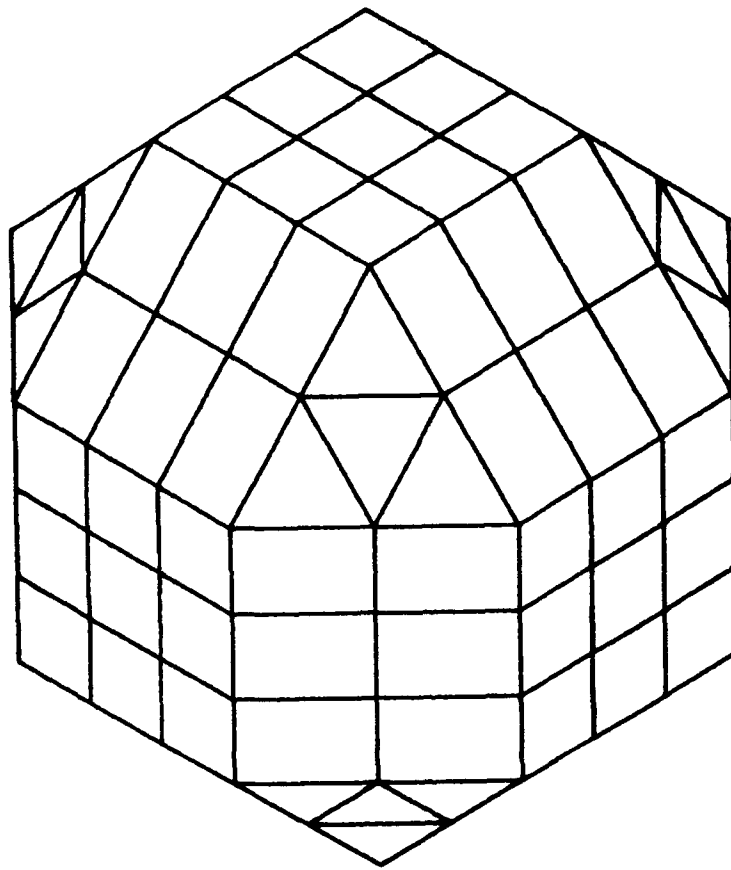


Figure 6/10. QSPHERE.

6.10.22 FIL111

The following cards define a FIL111:

```
FIL111
CORNERLINE x y z x' y' z'
FACE KAPTON 1 -1 -1
ENDOBJ
```

Notes:

1. FIL111: is the building block keyword.
2. CORNERLINE x y z x' y' z': defines both the length and the direction of the "step" FIL111 is to fill. The line must lie in one of the axis planes (XY, XZ, YZ) and must have a direction lying 45° to two of the axes. This means that one pair of the coordinates (x', x), (y, y') (z, z') must be identical and the other two pairs must differ by the same magnitude. For example,

```
'CORNERLINE 1 2 3 4 5 6'
```

is unacceptable since all three coordinate pairs change. The following correct example

```
'CORNERLINE 1 2 3 -1 4 3'
```

defines a line in the XY plane (Z is constant) with $\Delta x = -2$, and $\Delta y = +2$. Hence the line is 2.2 units in length and runs at 45° between the positive Y axis and the negative X axis.

3. FACE KAPTON 1 -1 -1: assigns the material KAPTON to the exposed surfaces of the FIL111 and defines its orientation via the surface normal of its exposed face: 1 -1 -1. The surface normal can only be combinations of

$\pm 1 \pm 1 \pm 1$

Only certain choices of corner line direction are consistent with each choice of FACE normal. If we subtract the x y z, x' y' z' coordinates defined in corner line

$$\Delta x = x' - x$$

$$\Delta y = y' - y$$

$$\Delta z = z' - z$$

then the surface normal $n_1 n_2 n_3$ (e.g., 1 1 1) must be orthogonal to $\Delta x, \Delta y, \Delta z$, i.e.,

$$\Delta x \quad n_1 + \Delta y \quad n_2 + \Delta z \quad n_3 = 0.$$

With the choice 1 2 3 -1 4 3 for the corner line coordinates only 1 1 +1 or -1 -1 +1 faces are permissible, e.g.,

$$-2. -1 + 2. -1 + 0. +1 = 0.$$

However, with -1 +1 +1

$$-2. -1 + 2.1 + 0. +1 = 4$$

the vectors are not orthogonal and so are not allowed.

As an example, the following cards:

```
FIL111
CORNERLINE 1 4 -6 1 7 -3
FACE GOLD -1 1 -1
ENDOBJ
```

defines a FIL111 covered with gold smoothing a step with a corner line running from 1 4 -6 in the YZ plane, between the positive Y and Z axis to 1 7 -3. The face of the FIL111 points in the negative X and Z directions and positive Y direction. (See Figure 6/11.)

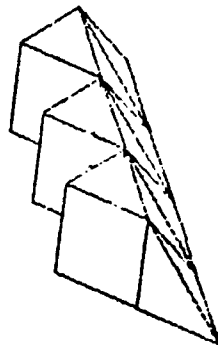


Figure 6/11. FIL111.

6.10.23 PLATE

The following cards define a PLATE:

```

PLATE
CORNER  x  y  z
DELTAS  Δx Δy Δz

TOP      ±  $\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$   ALUMIN

BOTTOM   ±  $\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$   KAPTON

ENDOBJ

```

Notes:

1. PLATE: is the building block keyword.
2. CORNER x y z: defines the vertex of the thin plate with the lowest indices (see 6.10.15, note 2).
3. DELTAS Δx Δy Δz: defines the length of the plate along the three axis directions. A PLATE may be thought of as a cuboid (or RECTAN) (see 6.10.15) with zero thickness in one direction. Hence one of Δx, Δy and Δz must be zero. For example, if Δy is chosen to be zero the PLATE will lie in the xz plane.

4. TOP ± $\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$ ALUMIN:

assigns the material ALUMIN to the TOP surface of the plate. The "TOP" surface may be either in a + or - axis direction. This choice is arbitrary unless a "double point" conflict is possible. Double point conflicts are explained in Section 6.11.11.

5. BOTTOM ± $\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$ KAPTON:

assigns the material KAPTON to the other side of the plate. If "top" were chosen as +X then bottom must be -X, and so on. Note that the choice of x, y or z must coincide with the Δx, Δy or Δz chosen to be zero.

As an example, the cards

```
PLATE  
CORNER 0 0 0  
DELTAS 0 2 2  
TOP -X TEFLON  
BOTTOM +X GOLD  
ENDOBJ
```

defines a 2 x 2 thin plate with gold on the +X side and teflon on the -X side lying in the YZ plane. (See Figure 6/12.)

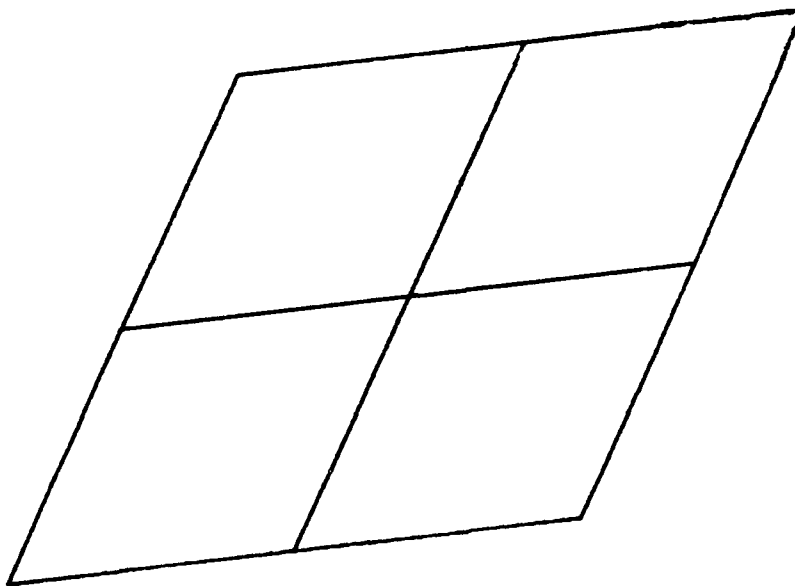


Figure 6/12. PLATE.

6.10.24 SLANT

The SLANT object should be thought of as a WEDGE with the 100-type surfaces left undefined. The FACE card is transformed to a TOP card, and a BOTTOM card is added. The CORNER is remote from the slanted plate, and is where the CORNER of a WEDGE would be if we were defining the FACE of a WEDGE. Similarly, the LENGTH card corresponds to that of a WEDGE. The syntax is

```
SLANT
CORNER  ix  jy  kz
TOP     matl nx  ny  nz
BOTTOM  matl
LENGTH  lx  ly  lz
ENDOBJ
```

6.10.25 MORE OBJECT DEFINITION KEYWORDS

In addition to the building block keywords and their parameter cards, the object definition of VEHICL also recognizes a few other keywords. With these and the building blocks, a complete object definition file can be constructed. Let us examine the remaining keywords and their effect one by one.

ENDSAT

Just as ENDOBJ terminates a set of building block parameter cards, so the keyword 'ENDSAT' terminates the whole object definition file. After reading an 'ENDSAT' card VEHICL stops reading from the object definition file and begins to process the information it has. Note that ALL object definition files must end with an 'ENDSAT' card.

COMMENT

VEHICL ignores anything written on the same 80 character line (or card) that begins with the keyword 'COMMENT'. This allows the user to include notes or reminders in long and complicated object definition files, e.g.,

```

•
•
•
COMMENT DEFINE OCTAGONAL BODY
OCTAGON
AXIS -6 0 2 -8 0 2
•
•
•

```

OFFSET

POLAR uses a coordinate system defined so that the lower, leftmost corner of the object definition grid has the coordinates (1,1,1). This is done so that all elements which contain the object will have addresses which are positive and nonzero for bit packing purposes.

Usually the computational grid coordinates are not convenient for object definition. The OFFSET card allows the user to shift the reference point during object definition. By default, the center of the object grid is labeled (0,0,0) for satellite description. The center of the grid in calculation coordinates is calculated by

$$\begin{aligned}
 x_c &= \text{int}[(NX+1)/2] \\
 y_c &= \text{int}[(NY+1)/2] \\
 z_c &= \text{int}[(NZ+1)/2]
 \end{aligned}$$

where NX, NY, and NZ are the object grid sizes as defined using the NXYZ keyword (6.21) in the normal VEHICL input mode. Point

(x_c, y_c, z_c) would then become the default $(0,0,0)$ for the duration of satellite definition.

The card

OFFSET a b c

would shift the coordinate system so that origin for satellite definition became $(x_c - a, y_c - b, z_c - c)$ in computational grid coordinates. The coordinate shift used during input will be the default or the most recent offset. Each additional OFFSET card will be used as an offset from the default shift (x_c, y_c, z_c) .

CONDUCTOR

POLAR allows for both insulating and conducting materials (Chapter 6.12). It assumes that all surface materials cover an underlying conductor. Up to 15 separate conductors are allowed. Each building block is associated with a particular conductor. This association is made by preceding all building block definitions associated with the first conductor with the card:

CONDUCTOR 1

Similarly, blocks associated with a second conductor are preceded by the card

CONDUCTOR 2

and so on. If no CONDUCTOR card is included in the object definition file all building blocks will be associated with CONDUCTOR 1. In the same way any building blocks defined before VEHICL encounters a card

CONDUCTOR n ($n > 1$)

will be associated with conductor 1. All subsequent blocks will be associated with conductor n, until another conductor card is encountered.

It is conventional to choose conductor 1 as the satellite ground conductor. Skipping conductor numbers is not recommended.

DELETE

DELETE allows the user to modify building blocks already defined by selectively "deleting" filled or partially filled cells (i.e., "deleting" them by making them empty).

DELETE x y z

empties the filled cell with the indices of its lowest index vertex given by x y z. (The lowest index vertex is the one with the sum of its X, Y and Z coordinates equal to the least positive number.) The coordinates x, y, z refer to the coordinate system presently active (i.e., the default system or that associated with the most recent OFFSET command). The DELETE command requires great care in its use. It does not assign materials to surfaces that are newly exposed by the removal of a filled element. The user must do this by defining a new object or objects with surfaces that coincide with those newly exposed. This is most easily done by overlaying objects (6.14).

COMPRESS

Since there is a limit of 1250 to the number of surfaces on an object (the rectangle defined in Figure 6/5 has 62 surfaces), large complex objects sometimes contain interior surfaces which need to be removed. Normally these surfaces are removed when the satellite definition is complete or when the number of surfaces exceeds the surface limit between building blocks, COMPRESS forces the existing interior surfaces to be removed immediately. An example of the syntax is

```

      •
      •
ENDOBJ
COMPRESS
RECTAN
      •
      •

```

MARKTB

Sometimes when defining odd objects with unusual geometries, undefined double points will be found. One way to patch the object is to use the MARKTB command to define an element and its associated surfaces to be TOPs or BOTTOMs.

```
MARKTB 1 2 2 TOP  
MARKTB 1 2 1 BOTTOM
```

The first MARKTB command forces the cell with lowest index vertices of (1,2,2) to be marked as a TOP element. The second line marks the element below the first in the Z-direction as a BOTTOM cell. It should be noted that these index vertices will be affected by previous OFFSET commands. (See Section 6.11.11 for more hints for dealing with double points.)

OTHER WORDS

Any other words that VEICL reads in the object definition file are assumed to be the names of new materials and VEICL then expects three more cards defining the material properties (6.12) to follow immediately.

6.11 DEFINING AN OBJECT: AN EXAMPLE

The input file of Figure 6/13 defines an object consisting of an ALUMINUM slab, trimmed with four KAPTON wedges and four TEFLON tetrahedra, and topped with a GOLD sphere. Three views of the resulting object are shown in Figure 6/14.

1.	COMMENT ALUMINUM SLAB	
2.	RECTAN	
3.	CORNER -3 -4 -1	
4.	DELTA 6 6 1	
5.	SURFACE +2 ALUMINUM	
6.	SURFACE -2 ALUMINUM	
7.	ENDOBJ	
8.	COMMENT FOUR KAPTON WEDGES	
9.	WEDGE	
10.	CORNER -3 -4 -1	
11.	FACE KAPTON -1 0 1	
12.	LENGTH 1 8 1	
13.	SURFACE -2 KAPTON	
14.	ENDOBJ	
15.	WEDGE	
16.	CORNER -3 -4 -1	
17.	FACE KAPTON 0 -1 1	
18.	LENGTH 6 1 1	
19.	SURFACE -2 KAPTON	
20.	ENDOBJ	
21.	WEDGE	
22.	CORNER 3 -4 -1	
23.	FACE KAPTON 1 0 1	
24.	LENGTH 1 8 1	
25.	SURFACE -2 KAPTON	
26.	ENDOBJ	
27.	WEDGE	
28.	CORNER -3 4 -1	
29.	FACE KAPTON 0 1 1	
30.	LENGTH 6 1 1	
31.	SURFACE -2 KAPTON	
32.	ENDOBJ	
33.	COMMENT FOUR TEFLON TETRAHEDRA	
34.	TETRAHEDRON	
35.	CORNER -3 -4 -1	
36.	FACE TEFLON -1 -1 1	
37.	LENGTH 1	
38.	SURFACE -2 TEFLON	
39.	ENDOBJ	
40.	TETRAHEDRON	
41.	CORNER 3 -4 -1	
42.	FACE TEFLON 1 -1 1	
43.	LENGTH 1	
44.	SURFACE -2 TEFLON	
45.	ENDOBJ	
46.	TETRAHEDRON	
47.	CORNER 3 4 -1	
48.	FACE TEFLON 1 1 1	
49.	LENGTH 1	
50.	SURFACE -2 TEFLON	
51.	ENDOBJ	
52.	TETRAHEDRON	
53.	CORNER -3 4 -1	
54.	FACE TEFLON -1 1 1	
55.	LENGTH 1	
56.	SURFACE -2 TEFLON	
57.	ENDOBJ	
58.	COMMENT ALL TOPPED BY A GOLD SPHERE	
59.	SPHERE	
60.	CENTER 0 0 2	
61.	DIAMETER 4	
62.	SIDE 2	
63.	MATERIAL GOLD	
64.	ENDOBJ	
65.	ENDSAT	

CUBOID

FOUR WEDGES

FOUR
TETRAHEDRA

SPHERE

Figure 6/13. Object definition example.

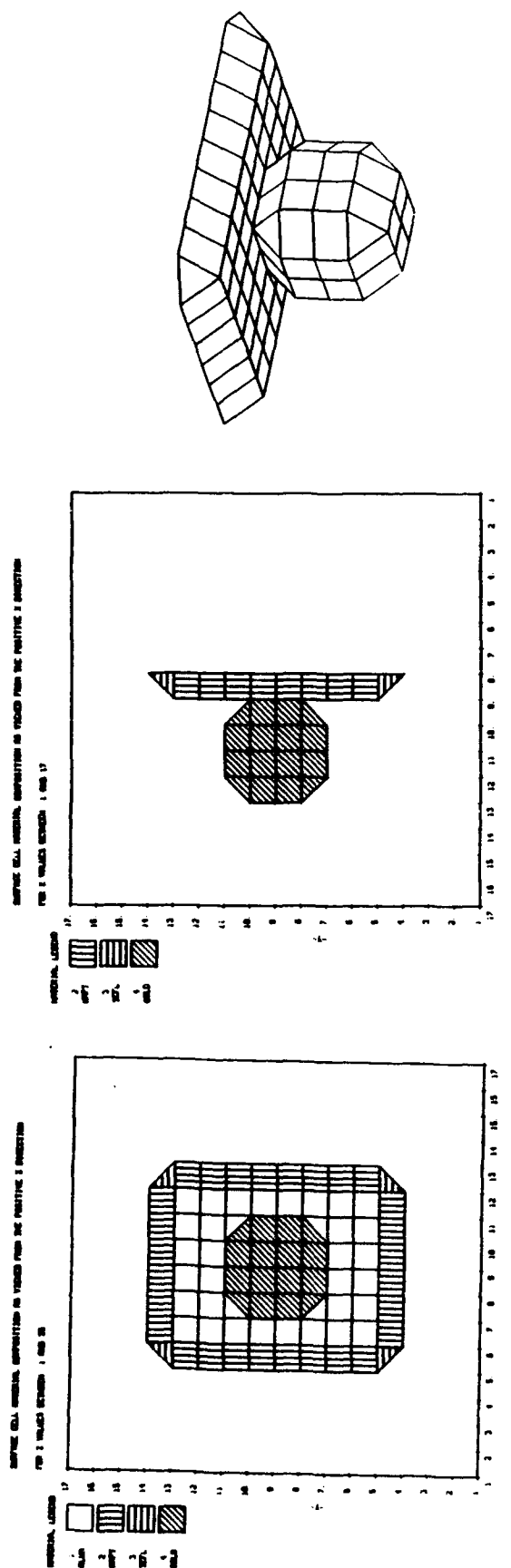


Figure 6/14. Three views of object defined by input of Figure 6/13.

6.11.10 LIMITATIONS IN OBJECT DEFINITION

It is probably fair to say that you can link building blocks together and nine times out of ten there will not be a problem. This section deals with the other one time out of ten, when what appears to be a perfectly reasonable combination of building blocks is rejected by VEHICL. We itemize here a rather formidable list of object definition "don'ts". However, you should remember that it takes hard work to break more than one or two of these rules defining any one object if you use a little common sense.

1. All exposed surfaces must be assigned materials.
2. The parameter cards for each building block, discussed in Section 6.10.14, must appear in the order shown, and no other.
3. The object must not touch the object grid boundary planes at any point.
4. Thin plates sharing the same volume element can do so only if the TOP face of one shares volume with the TOP of the other, or the BOTTOM face of one shares volume with the BOTTOM face of the other. TOP faces may not share volume elements with BOTTOM faces.
5. Thin plates may only intersect each other at the edges or corners.
6. Double points must be assigned TOP and BOTTOM sets (see Section 6.11.11).

(Rules 4 through 6 are all manifestations of conflicts involving double and triple points.)

6.11.11 DOUBLE POINTS

Thin plates may have different potentials on their two surfaces, yet they occupy only one plane of grid points. These grid points must therefore be associated with two distinct sets of potentials. For this reason they are called double points. The two sets of potentials associated with each half of the double points are distinguished by calling one set 'TOP' and one set 'BOTTOM'. Recall (6.10.23) that the

surfaces of a thin plate may be defined as 'TOP' or 'BOTTOM' regardless of whether their surface normal points along a positive or negative axis direction: The TOP and BOTTOM definition refers to the (arbitrary) choice of which set of potentials (TOP or BOTTOM) to associate with each surface. When double points share a volume element they must all be of the same type; i.e., all TOP or all BOTTOM. This is the basis for rule 4 in Section 6.11.10.

Double points also occur when other building blocks touch in such a way that their single points come together to form a common vertex of two "disjoint" volume elements. By "disjoint" volume elements we mean elements physically separated from each other by solid surfaces. This is shown for two cuboids touching along one edge only in Figure 6/15. The row of points along the touching edges are double points and one set must be defined as BOTTOM. This may be done by defining a thin plate touching the common edge. If the exterior surface of the plate pointing into one of the disjoint volumes is 'BOTTOM' then the half of the double point associated with the other disjoint volume becomes 'TOP'.

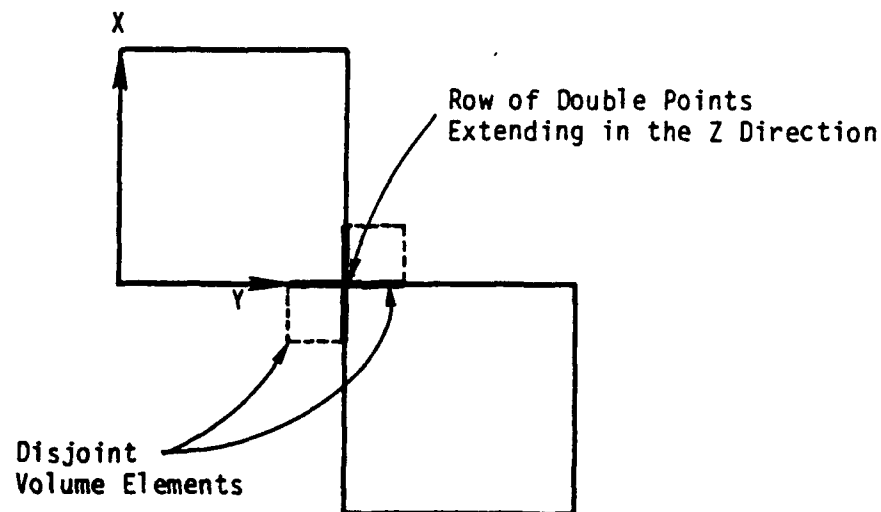


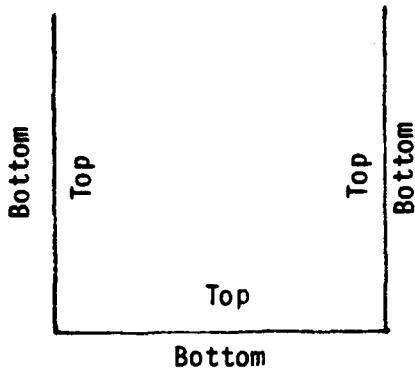
Figure 6/15. Profile of two cuboids sharing a common edge and resultant double points. Heavy lines show possible orientations for the definition of a thin plate to resolve the conflict.

Because of the way surface cell potentials are assigned to grid points, the edges of thin plates are only single points. However, a thin plate touching another building block with its edge creates a row of double points similar to that caused by two cuboids touching at an edge (Figure 6/16). These double points are automatically assigned TOP and BOTTOM sets.

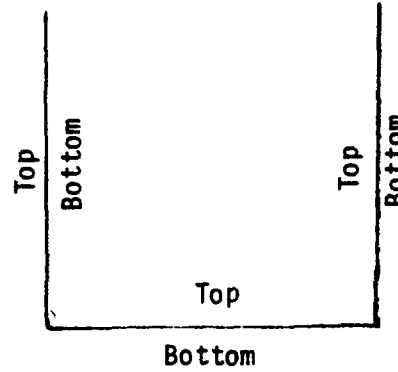
Double point ambiguities can also be resolved on an element by element basis using the MARKTB command discussed in Section 6.10.25.

6.11.12 TRIPLE POINTS

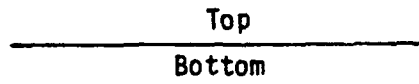
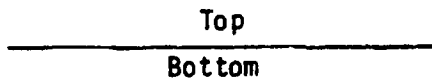
A triple point is said to occur when a vertex is common to three or more disjoint volume elements. Triple points are illegal! The easiest way to get a triple point is to define one thin plate passing through another. This is not allowed (rule 5, Section 6.11.10).



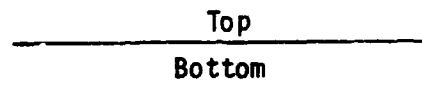
VALID



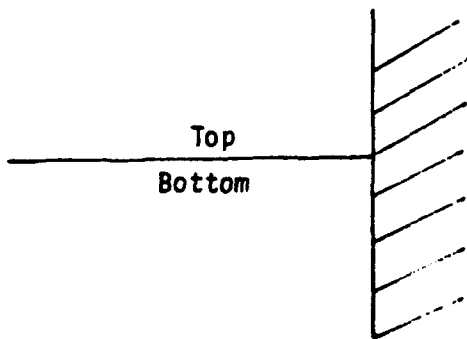
INVALID



VALID



INVALID



VALID

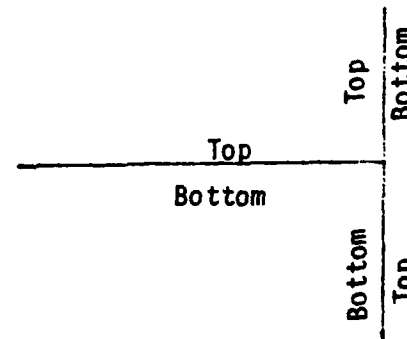
INVALID (Contains
triple point)

Figure 6/16. Examples of plates intersecting objects.

6.12 SURFACE MATERIALS

6.12.10 MATERIAL PROPERTIES

Each material name (e.g., KAPTON, GOLD, FRED (the name is arbitrary)) has associated with it a list of material properties. The name of each material and the values for each material property are supplied by the user in the object definition file. (This is explained in Section 6.12.11.) The nineteen material properties are summarized in Table 6/5. Here we examine each one in more detail.

DIELECTRIC CONSTANT (PROPERTY 1)

Property 1 contains the relative dielectric constant for an insulating material ϵ_r

$$\epsilon_r = \frac{\epsilon}{\epsilon_0}$$

where ϵ is the absolute dielectric constant and ϵ_0 is the dielectric constant of free space. ϵ_r is dimensionless.

THICKNESS (PROPERTY 2)

Property 2 gives the thickness d of a dielectric film covering an underlying conductor in meters. d is arbitrary and may be chosen to be more or less than a mesh unit. However, note that POLAR uses thin-film approximations in many of its calculations involving surfaces (Section 4.52).

BULK CONDUCTIVITY (PROPERTY 3)

Property 3 gives the bulk conductivity σ_0 of the surface material in $\text{ohms}^{-1} \text{m}^{-1}$. σ_0 is assumed to be the value appropriate for a sample not exposed to any radiation and not subject to any internal electric fields. Field enhancement and radiation enhancement of σ_0 are not currently modeled by POLAR. A value of -1 indicates that the material is a metallic conductor.

TABLE 6/5. MATERIAL PROPERTIES
(see Section 6.12.10 for notes)

<u>Property No.</u>	<u>User Input Units</u>	<u>Description</u>
1	None	Relative dielectric constant
2	m	Dielectric material thickness
3	ohms ⁻¹ m ⁻¹	Bulk conductivity (= -1 for a metallic conductor)
4	None	Atomic number
5	None	Maximum secondary electron yield for electron impact
6	keV	Primary electron energy that produces maximum secondary yield
7	angstroms	$\left\{ \begin{array}{l} \text{Range parameters (4.3)} \\ R = P_7 E^{P_8} + P_9 E^{P_{10}} \end{array} \right.$
8	None	
9	angstroms	
10	None	
11	None	Secondary electron yield due to impact of 1 keV protons
12	keV	Incident proton energy that produces maximum secondary electron yield
13	A m ⁻²	Photoelectron yield for normally incident sunlight
14	ohms square ⁻¹	Surface resistivity (= -1 for non-conducting surface)
15	Volts	Maximum (absolute) potential attainable before a discharge must occur
16	Volts	Maximum potential difference between surface and underlying conductor before a discharge must occur
17	ohms ⁻¹ m ⁻¹ (m ² s ⁻³) ⁻¹	Radiation-induced conductivity coefficient (k)
18	None	Radiation-induced conductivity power (Δ)
19	kg m ⁻³	Material density

ATOMIC NUMBER (PROPERTY 4)

Property 4 is the atomic number for pure elements or the mean atomic number for chemical compounds; e.g., polyethylene $(CH_2)_n$ has a mean atomic number of $(6 + 1 + 1)/3 = 2.7$.

SECONDARY YIELD (PROPERTIES 5 AND 6)

Properties 5 and 6 are the coordinates of the maximum in the secondary electron yield curve of the material. The secondary yield curve is a plot of secondary yield δ

$$\delta = \frac{\text{current of secondary electrons emitted}}{\text{incident primary electron current}}$$

for normally incident electrons, against the incident energy of the primary electron E . This is further discussed and illustrated in Section 4.52. Property 5 contains δ_{\max} , and property 6 contains E_{\max} in keV.

ELECTRON RANGE (PROPERTIES 7, 8, 9 AND 10)

Part of the secondary electron emission formulation requires an analytical form for the "range" of electrons in the material. The range is the depth to which the electrons can penetrate the material as they are continuously slowed down by losing energy to the material lattice. POLAR uses a biexponential form. If P_7 , P_8 , P_9 , and P_{10} are properties 7-10 respectively, the range R is given by

$$R = P_7 E^{P_8} + P_9 E^{P_{10}}$$

The four parameters are obtained from fits to stopping power data (Section 4.52). The range is determined in \AA (10^{-10} m). If no

reliable stopping power data or four parameter fits are available, the range may be estimated from Feldman's formula^[4] automatically by assigning -1 to property 7. In this mode properties 7-10 are assigned as follows:

$$P_7 = -1$$

$$P_8 = \text{null}$$

$$P_9 = \text{material density (g cm}^{-3}\text{)}$$

$$P_{10} = \text{mean atomic weight (AMU)}$$

The mean atomic weight is calculated in the same way as the mean atomic number (property 4) using atomic masses rather than numbers.

ION INDUCED SECONDARY EMISSION (PROPERTIES 11 AND 12)

Secondary emission of electrons due to ion impact is also treated using a two parameter theory (4.52). Parameter 11 contains the yield for 1 keV normally incident protons and parameter 12 the proton energy that produces the maximum electron yield. The secondary emission properties due to impact of ions other than protons are assumed to be identical to the proton values.

PHOTOEMISSION (PROPERTY 13)

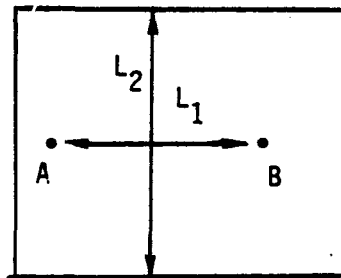
Property 13 contains the yield of photoelectrons from the surface material exposed to the solar spectrum. The intensity is that measured on earth 93,000,000 miles from the sun. (Earth orbit altitudes are negligible by comparison and the intensity of the sun close to earth may be considered constant.)

SURFACE RESISTIVITY (PROPERTY 14)

Property 14 gives the intrinsic surface resistivity in the "ohms per square". This rather odd unit is used to distinguish the resistivity coefficient (property 14) from the actual surface resistance (in ohms) calculated by POLAR. Consider two points in a plane A and B, a distance L_1 apart. If L_2 is the "width" of the plane

$$\text{surface resistance} = \text{surface resistivity} \times \frac{L_1}{L_2}$$

$$\text{i.e.} \quad \text{ohms} = (\text{ohms per square}) \times \begin{matrix} \text{dimensionless} \\ \text{geometrical} \\ \text{factor} \end{matrix}$$



POLAR uses the surface resistivity per square, times a geometrical factor it calculates to determine the surface resistance between two adjacent materials.

The intrinsic surface conductivity is due to the migration of electrons along the surface layer aided by adsorbed impurities and defects.

Surface conductivity may be omitted from the current calculations completely by choosing property 14 to be negative.

DISCHARGE ANALYSIS (PROPERTIES 17, 18, 19, 20)

Currently under development.

PROPERTIES 17, 18, 19, 20 (RADIATION INDUCED CONDUCTIVITY)

Currently under development.

6.12.11 DEFINING MATERIALS

New materials are defined, and their properties assigned inside the object definition file (6.10.11). The object definition file is read by POLAR module VEHICL. VEHICL interprets any word that it does not recognize as a building block keyword (or their parameter cards (6.10.25)) as the definition of a new material name. New material names may not appear inside building block definitions (i.e., between a building block keyword and an 'ENDOBJ' statement).

Following the material name, VEHICL expects to find three additional cards specifying 20 constants as the material properties to be associated with the name. The 20 constants correspond to properties 1-20 and are read sequentially; i.e., the first constant read is interpreted as property 1, the second, property 2, and so on. They are arranged sequentially, eight per card, so that cards 1 and 2 each have eight numbers and card 3, four numbers. Formally each number is written in a field of up to ten characters, but POLAR will read the cards in free format. No units need be specified. POLAR will assume the units given in Table 6/5 and no others. For properties not requiring any input such as property 20, or properties 17-19 for conductors, some constant must be entered but its value is arbitrary. (POLAR will not actually use the values entered but expects to read something.)

Once the three material property cards have been read VEHICL is ready to accept any other keywords or more material names. POLAR will recognize up to fifteen different materials.

Materials must be defined before they are referred to in any building block definition. For example, if I assign the surface of a sphere to be 'FSTUFF' with the card

MATERIAL FSTUFF

if 'FSTUFF' and its material properties have not been declared earlier in the object definition file, an error will occur and execution will stop. For this reason all the materials to be used are usually declared at the very beginning of the object definition file. This is shown in Figure 6/17.

COMMENT DEFINITION OF SATELLITE "BIG EARS"

Material Name 1

3 material property cards

Material Name 2

3 material property cards

•
•
•

COMMENT DEFINE MAIN BODY

CONDUCTOR 1

QSPHERE

parameter cards

ENDOBJ

RECTAN

parameter cards

ENDOBJ

•
• more building blocks
•

COMMENT DEFINE SOLAR PANEL (SEPARATE CONDUCTOR)

CONDUCTOR 2

PLATE

parameter cards

ENDOBJ

•
• more building blocks
•

COMMENT

CONDUCTOR 3

•
• more conductor segments
•

ENDSAT

Figure 6/17. General form of the object definition file.

6.12.12 DEFAULT MATERIALS

There is one case where the user can forget to define his or her materials and get away with it. When VEHICL encounters a material that hasn't been defined already, before an error occurs, it checks the following list of default materials:

ALUMIN
AQUADG
CPAINT
GOLD
INDOX
MAGNES
SCREEN
KAPTON
NPAINT
SI02
SOLAR
TEFLON
SILVER

If the material is included in this list, it becomes one of the up to fifteen defined materials and its properties, stored internally, are automatically entered as VEHICL input by the code. The properties of these materials are shown in Table 6/6. Any further reference to the material will assign the same set of properties to the surfaces concerned. If the material is not found in this list, an error will occur. These material properties are currently a carryover from NASCAP. New default materials will be included in future revisions.

If two sets of material properties are defined with the same name, or names with the same first four letters, two of the fifteen possible materials are used up but only the first set of material properties are used. For example, if GOLD is referenced before it is defined in the runstream, the default material properties of gold will be associated with all gold surfaces in the object definition file.

TABLE 6/6
MATERIAL PROPERTIES

MATERIAL 1: ALUMIN

PROPERTY	INPUT VALUE	CODE VALUE
1 DIELECTRIC CONSTANT	1.00+000 (NONE)	1.00+000 (NONE)
2 THICKNESS	1.00+003 METERS	1.00+002 MESH
3 CONDUCTIVITY	-1.00+000 MHC/M	-1.00+000 MHC/M
4 ATOMIC NUMBER	1.00+001 (NONE)	1.00+001 (NONE)
5 DELTA MAX >COEFF	9.70+001 (NONE)	9.18+000 (NONE)
6 E-MAX >DEPTH**=1	3.00+001 KEV	3.00+002 ANG-01
7 RANGE	1.54+002 ANG.	1.23+002 ANG.
8 EXPONENT > RANGE	9.00+001 (NONE)	3.87+002 ANG.
9 RANGE > EXPONENT	2.20+002 ANG.	8.00+001 (NONE)
10 EXPONENT	1.76+000 (NONE)	1.76+000 (NONE)
11 YIELD FOR 1KEV PROTONS	2.44+001 (NONE)	2.44+001 (NONE)
12 MAX DE/DX FOR PROTONS	2.30+002 KEV	2.30+002 KEV
13 PHOTOCURRENT	4.00+005 A/M**2	4.00+005 A/M**2
14 SURFACE RESISTIVITY	-1.00+000 OHMS	-9.85+013 V-S/C
15 SPACE DISCHARGE POT'L	1.00+004 VOLTS	1.00+004 VOLTS
16 INTERNAL DISCHARGE POT'L	2.00+003 VOLTS	2.00+003 VOLTS
17 RADN INDUCEDCOND*YCOEFF	1.00+013 MHOMS3	1.00+013 MHOMS3
18 RADN INDUCEDCOND*YPOWER	1.00+000 (NONE)	1.00+000 (NONE)
19 DENSITY	1.00+003 KG/M**3	1.00+003 KG/M**3
20	2.00+001	-1.00+000

MATERIAL 2: AQUAD

PROPERTY	INPUT VALUE	CODE VALUE
1 DIELECTRIC CONSTANT	1.00+000 (NONE)	1.00+000 (NONE)
2 THICKNESS	1.00+003 METERS	1.00+002 MESH
3 CONDUCTIVITY	-1.00+000 MHC/M	-1.00+000 MHC/M
4 ATOMIC NUMBER	0.00+000 (NONE)	0.00+000 (NONE)
5 DELTA MAX >COEFF	1.00+000 (NONE)	7.06+000 (NONE)
6 E-MAX >DEPTH**=1	3.00+001 KEV	2.21+002 ANG-01
7 RANGE	-1.00+000 ANG.	5.80+002 ANG.
8 EXPONENT > RANGE	.00 (NONE)	.00 ANG.
9 RANGE > EXPONENT	2.00+000 ANG.	1.55+000 (NONE)
10 EXPONENT	1.20+001 (NONE)	1.00+000 (NONE)
11 YIELD FOR 1KEV PROTONS	4.55+001 (NONE)	4.55+001 (NONE)
12 MAX DE/DX FOR PROTONS	1.40+002 KEV	1.40+002 KEV
13 PHOTOCURRENT	2.10+005 A/M**2	2.10+005 A/M**2
14 SURFACE RESISTIVITY	-1.00+000 OHMS	-8.85+013 V-S/C
15 SPACE DISCHARGE POT'L	1.00+004 VOLTS	1.00+004 VOLTS
16 INTERNAL DISCHARGE POT'L	2.00+003 VOLTS	2.00+003 VOLTS
17 RADN INDUCEDCOND*YCOEFF	1.00+013 MHOMS3	1.00+013 MHOMS3
18 RADN INDUCEDCOND*YPOWER	1.00+000 (NONE)	1.00+000 (NONE)
19 DENSITY	1.00+003 KG/M**3	1.00+003 KG/M**3
20	2.00+001	-1.00+000

TABLE 6/6
MATERIAL PROPERTIES (Continued)

MATERIAL 3: CPAINT

PROPERTY	INPUT VALUE	CODE VALUE
1 DIELECTRIC CONSTANT	3.50+000 (NONE)	3.50+000 (NONE)
2 THICKNESS	1.00+003 METERS	1.00+002 MESH
3 CONDUCTIVITY	-1.00+000 MH0/M	-1.00+000 MH0/M
4 ATOMIC NUMBER	5.00+000 (NONE)	5.00+000 (NONE)
5 DELTA MAX >COEFF	2.10+000 (NONE)	4.06+001 (NONE)
6 E-MAX >DEPTH**1	1.50+001 KEV	8.74+002 ANG-01
7 RANGE	7.15+001 ANG.	4.29+001 ANG.
8 EXPONENT > RANGE	6.00+001 (NONE)	5.52+002 ANG.
9 RANGE > EXPONENT	3.12+002 ANG.	6.00+001 (NONE)
10 EXPONENT	1.77+000 (NONE)	1.77+000 (NONE)
11 YIELD FOR 1KEV PROTONS	4.55+001 (NONE)	4.55+001 (NONE)
12 MAX DE/0X FOR PROTONS	1.40+002 KEV	1.40+002 KEV
13 PHOTOCURRENT	2.00+005 A/M**2	2.00+005 A/M**2
14 SURFACE RESISTIVITY	-1.00+000 OHMS	-8.85+013 V-S/Q
15 SPACE DISCHARGE POT'L	1.00+004 VOLTS	1.00+004 VOLTS
16 INTERNAL DISCHARGE POT'L	2.00+003 VOLTS	2.00+003 VOLTS
17 RADN INDUCED COND*YCOEFF	1.00+013 MHOM/S	1.00+013 MHOM/S
18 RADN INDUCED COND*YPOWER	1.00+000 (NONE)	1.00+000 (NONE)
19 DENSITY	1.00+003 KG/M**3	1.00+003 KG/M**3
20	2.00+001	-1.00+000

MATERIAL 4: GOLD

PROPERTY	INPUT VALUE	CODE VALUE
1 DIELECTRIC CONSTANT	1.00+000 (NONE)	1.00+000 (NONE)
2 THICKNESS	1.00+003 METERS	1.00+002 MESH
3 CONDUCTIVITY	-1.00+000 MH0/M	-1.00+000 MH0/M
4 ATOMIC NUMBER	7.90+001 (NONE)	7.90+001 (NONE)
5 DELTA MAX >COEFF	8.60+001 (NONE)	2.93+000 (NONE)
6 E-MAX >DEPTH**1	6.00+001 KEV	2.02+002 ANG-01
7 RANGE	8.88+001 ANG.	8.17+001 ANG.
8 EXPONENT > RANGE	9.20+001 (NONE)	9.25+001 ANG.
9 RANGE > EXPONENT	5.35+001 ANG.	9.20+001 (NONE)
10 EXPONENT	1.73+000 (NONE)	1.73+000 (NONE)
11 YIELD FOR 1KEV PROTONS	4.13+001 (NONE)	4.13+001 (NONE)
12 MAX DE/0X FOR PROTONS	1.35+002 KEV	1.35+002 KEV
13 PHOTOCURRENT	2.90+005 A/M**2	2.90+005 A/M**2
14 SURFACE RESISTIVITY	-1.00+000 OHMS	-8.85+013 V-S/Q
15 SPACE DISCHARGE POT'L	1.00+004 VOLTS	1.00+004 VOLTS
16 INTERNAL DISCHARGE POT'L	2.00+003 VOLTS	2.00+003 VOLTS
17 RADN INDUCED COND*YCOEFF	1.00+013 MHOM/S	1.00+013 MHOM/S
18 RADN INDUCED COND*YPOWER	1.00+000 (NONE)	1.00+000 (NONE)
19 DENSITY	1.90+003 KG/M**3	1.90+003 KG/M**3
20	2.00+001	-1.00+000

TABLE 6/6
MATERIAL PROPERTIES (Continued)

MATERIAL 5: INDOX

	PROPERTY	INPUT VALUE	CODE VALUE
1	DIELECTRIC CONSTANT	1.00+000 (NONE)	1.00+000 (NONE)
2	THICKNESS	1.00+003 METERS	1.00+002 MESH
3	CONDUCTIVITY	-1.00+000 MH0/M	-1.00+000 MH0/M
4	ATOMIC NUMBER	2.44+001 (NONE)	2.44+001 (NONE)
5	DELTA MAX >COEFF	1.40+000 (NONE)	3.02+000 (NONE)
6	E-MAX >DEPTH**=1	8.00+001 KEV	1.49+002 ANG-01
7	RANGE	-1.00+000 ANG.	1.57+002 ANG.
8	EXPONENT > RANGE	.00 (NONE)	.00 ANG.
9	RANGE > EXPONENT	7.18+000 ANG.	2.01+000 (NONE)
10	EXPONENT	5.55+001 (NONE)	1.00+000 (NONE)
11	YIELD FOR 1KEV PROTONS	4.90+001 (NONE)	4.90+001 (NONE)
12	MAX DE/0X FOR PROTONS	1.23+002 KEV	1.23+002 KEV
13	PHOTOCURRENT	3.20+005 A/M**2	3.20+005 A/M**2
14	SURFACE RESISTIVITY	-1.00+000 OHMS	-9.85+013 V-S/C
15	SPACE DISCHARGE POT'L	1.00+004 VOLTS	1.00+004 VOLTS
16	INTERNAL DISCHARGE POT'L	2.00+003 VOLTS	2.00+003 VOLTS
17	RADN INDUCEDCOND*YCOEFF	1.00+013 MHOMS3	1.00+013 MHOMS3
18	RADN INDUCEDCOND*YPOWER	1.00+000 (NONE)	1.00+000 (NONE)
19	DENSITY	1.00+003 KG/M*3	1.00+003 KG/M*3
20		2.00+001	-1.00+000

MATERIAL 6: MAGNES

	PROPERTY	INPUT VALUE	CODE VALUE
1	DIELECTRIC CONSTANT	1.00+000 (NONE)	1.00+000 (NONE)
2	THICKNESS	1.00+003 METERS	1.00+002 MESH
3	CONDUCTIVITY	-1.00+000 MH0/M	-1.00+000 MH0/M
4	ATOMIC NUMBER	1.20+001 (NONE)	1.00+001 (NONE)
5	DELTA MAX >COEFF	9.00+001 (NONE)	7.02+000 (NONE)
6	E-MAX >DEPTH**=1	2.50+001 KEV	2.79+002 ANG-01
7	RANGE	-1.00+000 ANG.	6.96+002 ANG.
8	EXPONENT > RANGE	.00 (NONE)	.00 ANG.
9	RANGE > EXPONENT	1.74+000 ANG.	1.75+000 (NONE)
10	EXPONENT	2.43+001 (NONE)	1.00+000 (NONE)
11	YIELD FOR 1KEV PROTONS	2.44+001 (NONE)	2.44+001 (NONE)
12	MAX DE/0X FOR PROTONS	2.30+002 KEV	2.30+002 KEV
13	PHOTOCURRENT	4.00+005 A/M**2	4.00+005 A/M**2
14	SURFACE RESISTIVITY	-1.00+000 OHMS	-8.85+013 V-S/C
15	SPACE DISCHARGE POT'L	1.00+004 VOLTS	1.00+004 VOLTS
16	INTERNAL DISCHARGE POT'L	2.00+003 VOLTS	2.00+003 VOLTS
17	RADN INDUCEDCOND*YCOEFF	1.00+013 MHOMS3	1.00+013 MHOMS3
18	RADN INDUCEDCOND*YPOWER	1.00+000 (NONE)	1.00+000 (NONE)
19	DENSITY	1.00+003 KG/M*3	1.00+003 KG/M*3
20		2.00+001	-1.00+000

TABLE 6/6
MATERIAL PROPERTIES (Continued)

MATERIAL 7: SCREEN

PROPERTY	INPUT VALUE	CODE VALUE
1 DIELECTRIC CONSTANT	1.00+000 (NONE)	1.00+000 (NONE)
2 THICKNESS	1.00+003 METERS	1.00+002 ME'
3 CONDUCTIVITY	-1.00+000 MHO/M	-1.00+000 MHO/M
4 ATOMIC NUMBER	1.00+000 (NONE)	1.00+000 (NONE)
5 DELTA MAX >COEFF	.00 (NONE)	.00 (NONE)
6 E-MAX >DEPTH**=1	1.00+000 KEV	1.00+001 ANG-C1
7 RANGE	1.00+001 ANG.	1.00+001 ANG.
8 EXPONENT > RANGE	1.00+000 (NONE)	.00 ANG.
9 RANGE > EXPONENT	.00 ANG.	1.00+000 (NONE)
10 EXPONENT	1.00+000 (NONE)	1.00+000 (NONE)
11 YIELD FOR 1KEV PROTONS	.00 (NONE)	.00 (NONE)
12 MAX DE/DOX FOR PROTONS	1.00+000 KEV	1.00+000 KEV
13 PHOTOCURRENT	.00 A/M**2	.00 A/M**2
14 SURFACE RESISTIVITY	-1.00+000 OHMS	-8.85-013 V-S/O
15 SPACE DISCHARGE POT'L	1.00+004 VOLTS	1.00+004 VOLTS
16 INTERNAL DISCHARGE POT'L	2.00+003 VOLTS	2.00+003 VOLTS
17 RADN INDUCEDCOND'YCOEFF	1.00+013 MHOMS3	1.00+013 MHOMS3
18 RADN INDUCEDCOND'YPOWER	1.00+000 (NONE)	1.00+000 (NONE)
19 DENSITY	1.00+003 KG/M*3	1.00+003 KG/M*3
20	2.00+001	-1.00+000

MATERIAL 8: KAPTON

PROPERTY	INPUT VALUE	CODE VALUE
1 DIELECTRIC CONSTANT	3.50+000 (NONE)	3.50+000 (NONE)
2 THICKNESS	1.27-004 METERS	1.27-003 MESH
3 CONDUCTIVITY	1.00-016 MHO/M	1.00-016 MHO/M
4 ATOMIC NUMBER	5.00+000 (NONE)	5.00+000 (NONE)
5 DELTA MAX >COEFF	2.10+000 (NONE)	4.06+001 (NONE)
6 E-MAX >DEPTH**=1	1.00+001 KEV	8.74-002 ANG-C1
7 RANGE	7.15+001 ANG.	4.29+001 ANG.
8 EXPONENT > RANGE	6.00+001 (NONE)	5.52+002 ANG.
9 RANGE > EXPONENT	3.12+002 ANG.	6.00+001 (NONE)
10 EXPONENT	1.77+000 (NONE)	1.77+000 (NONE)
11 YIELD FOR 1KEV PROTONS	4.55-001 (NONE)	4.55-001 (NONE)
12 MAX DE/DOX FOR PROTONS	1.40+002 KEV	1.40+002 KEV
13 PHOTOCURRENT	2.00+005 A/M**2	2.00+005 A/M**2
14 SURFACE RESISTIVITY	1.00+016 OHMS	8.85+003 V-S/O
15 SPACE DISCHARGE POT'L	1.00+004 VOLTS	1.00+004 VOLTS
16 INTERNAL DISCHARGE POT'L	2.00+003 VOLTS	2.00+003 VOLTS
17 RADN INDUCEDCOND'YCOEFF	1.00+013 MHOMS3	1.00+013 MHOMS3
18 RADN INDUCEDCOND'YPOWER	1.00+000 (NONE)	1.00+000 (NONE)
19 DENSITY	1.00+003 KG/M*3	1.00+003 KG/M*3
20	2.00+001	1.00+016

TABLE 6/6
MATERIAL PROPERTIES (Continued)

MATERIAL 9: NPAINT

PROPERTY	INPUT VALUE	CODE VALUE
1 DIELECTRIC CONSTANT	3.50+000 (NONE)	3.50+000 (NONE)
2 THICKNESS	5.00+005 METERS	5.00+004 MESH
3 CONDUCTIVITY	5.90+014 MHO/M	5.90+014 MHO/M
4 ATOMIC NUMBER	5.00+000 (NONE)	5.00+000 (NONE)
5 DELTA MAX >COEFF	2.10+000 (NONE)	3.05+001 (NONE)
6 E-MAX >DEPTH**1	1.50+001 KEV	1.41+002 ANG-C1
7 RANGE	-1.00+000 ANG.	1.05+003 ANG.
8 EXPONENT > RANGE	.00 (NONE)	.00 ANG.
9 RANGE > EXPONENT	1.05+000 ANG.	1.51+000 (NONE)
10 EXPONENT	9.80+000 (NONE)	1.00+000 (NONE)
11 YIELD FOR 1KEV PROTONS	4.55+001 (NONE)	4.55+001 (NONE)
12 MAX DE/DX FOR PROTONS	1.40+002 KEV	1.40+002 KEV
13 PHOTOCURRENT	2.00+005 A/M**2	2.00+005 A/M**2
14 SURFACE RESISTIVITY	1.00+013 OHMS	8.85+000 V-S/C
15 SPACE DISCHARGE POT'L	1.00+004 VOLTS	1.00+004 VOLTS
16 INTERNAL DISCHARGE POT'L	2.00+003 VOLTS	2.00+003 VOLTS
17 RADN INDUCEDCOND*YCOEFF	1.00+013 MHOMS3	1.00+013 MHOMS3
18 RADN INDUCEDCOND*YPOWER	1.00+000 (NONE)	1.00+000 (NONE)
19 DENSITY	1.00+003 KG/M*3	1.00+003 KG/M*3
20	2.00+001	5.90+014

MATERIAL 10: SIO2

PROPERTY	INPUT VALUE	CODE VALUE
1 DIELECTRIC CONSTANT	4.00+000 (NONE)	4.00+000 (NONE)
2 THICKNESS	1.27+004 METERS	1.27+003 MESH
3 CONDUCTIVITY	1.00+014 MHO/M	1.00+014 MHO/M
4 ATOMIC NUMBER	1.00+001 (NONE)	1.00+001 (NONE)
5 DELTA MAX >COEFF	2.40+000 (NONE)	1.46+001 (NONE)
6 E-MAX >DEPTH**1	4.00+001 KEV	2.21+002 ANG-D1
7 RANGE	1.16+002 ANG.	9.42+001 ANG.
8 EXPONENT > RANGE	8.10+001 (NONE)	3.41+002 ANG.
9 RANGE > EXPONENT	1.83+002 ANG.	8.10+001 (NONE)
10 EXPONENT	1.86+000 (NONE)	1.86+000 (NONE)
11 YIELD FOR 1KEV PROTONS	4.55+001 (NONE)	4.55+001 (NONE)
12 MAX DE/DX FOR PROTONS	1.40+002 KEV	1.40+002 KEV
13 PHOTOCURRENT	2.00+005 A/M**2	2.00+005 A/M**2
14 SURFACE RESISTIVITY	1.00+019 OHMS	8.85+006 V-S/C
15 SPACE DISCHARGE POT'L	1.00+004 VOLTS	1.00+004 VOLTS
16 INTERNAL DISCHARGE POT'L	2.00+003 VOLTS	2.00+003 VOLTS
17 RADN INDUCEDCOND*YCOEFF	1.00+013 MHOMS3	1.00+013 MHOMS3
18 RADN INDUCEDCOND*YPOWER	1.00+000 (NONE)	1.00+000 (NONE)
19 DENSITY	1.00+003 KG/M*3	1.00+003 KG/M*3
20	2.00+001	1.00+014

TABLE 6/6
MATERIAL PROPERTIES (Continued)

MATERIAL 11: SOLAR

PROPERTY	INPUT VALUE	CODE VALUE
1 DIELECTRIC CONSTANT	3.80+000 (NONE)	3.80+000 (NONE)
2 THICKNESS	1.79+004 METERS	1.79+003 MESH
3 CONDUCTIVITY	1.00+017 MHO/M	1.00+017 MHO/M
4 ATOMIC NUMBER	1.00+001 (NONE)	1.00+001 (NONE)
5 DELTA MAX >COEFF	2.05+000 (NONE)	1.31+001 (NONE)
6 E-MAX >DEPTH**=1	4.10+001 KEV	3.17+002 ANG-C1
7 RANGE	7.75+001 ANG.	3.49+001 ANG.
8 EXPONENT > RANGE	4.50+001 (NONE)	2.70+002 ANG.
9 RANGE > EXPONENT	1.56+002 ANG.	4.50+001 (NONE)
10 EXPONENT	1.73+000 (NONE)	1.73+000 (NONE)
11 YIELD FOR 1KEV PROTONS	2.44+001 (NONE)	2.44+001 (NONE)
12 MAX DE/DX FOR PROTONS	2.30+002 KEV	2.30+002 KEV
13 PHOTOCURRENT	2.00+005 A/M**2	2.00+005 A/M**2
14 SURFACE RESISTIVITY	1.00+019 OHMS	9.85+006 V-S/O
15 SPACE DISCHARGE POT'L	1.00+004 VOLTS	1.00+004 VOLTS
16 INTERNAL DISCHARGE POT'L	2.00+003 VOLTS	2.00+003 VOLTS
17 RADN INDUCEDCOND*YCJEFF	1.00+013 MHOMS3	1.00+013 MHOMS3
18 RADN INDUCEDCOND*YPO=EP	1.00+000 (NONE)	1.00+000 (NONE)
19 DENSITY	1.00+003 KG/M**3	1.00+003 KG/M**3
20	2.00+001	1.00+017

MATERIAL 12: TEFLON

PROPERTY	INPUT VALUE	CODE VALUE
1 DIELECTRIC CONSTANT	2.00+000 (NONE)	2.00+000 (NONE)
2 THICKNESS	1.27+004 METERS	1.27+003 MESH
3 CONDUCTIVITY	1.00+016 MHO/M	1.00+016 MHO/M
4 ATOMIC NUMBER	7.00+000 (NONE)	7.00+000 (NONE)
5 DELTA MAX >COEFF	3.00+000 (NONE)	2.27+001 (NONE)
6 E-MAX >DEPTH**=1	3.00+001 KEV	3.93+002 ANG-C1
7 RANGE	4.54+001 ANG.	1.81+001 ANG.
8 EXPONENT > RANGE	4.00+001 (NONE)	3.65+002 ANG.
9 RANGE > EXPONENT	2.18+002 ANG.	4.00+001 (NONE)
10 EXPONENT	1.77+000 (NONE)	1.77+000 (NONE)
11 YIELD FOR 1KEV PROTONS	4.55+001 (NONE)	4.55+001 (NONE)
12 MAX DE/DX FOR PROTONS	1.40+002 KEV	1.40+002 KEV
13 PHOTOCURRENT	2.00+005 A/M**2	2.00+005 A/M**2
14 SURFACE RESISTIVITY	1.00+016 OHMS	8.85+003 V-S/O
15 SPACE DISCHARGE POT'L	1.00+004 VOLTS	1.00+004 VOLTS
16 INTERNAL DISCHARGE POT'L	2.00+003 VOLTS	2.00+003 VOLTS
17 RADN INDUCEDCOND*YCJEFF	1.00+013 MHOMS3	1.00+013 MHOMS3
18 RADN INDUCEDCOND*YPOWER	1.00+000 (NONE)	1.00+000 (NONE)
19 DENSITY	1.00+003 KG/M**3	1.00+003 KG/M**3
20	2.00+001	1.00+016

TABLE 6/6
MATERIAL PROPERTIES (Concluded)

MATERIAL 13: SILVER

	PROPERTY	INPUT VALUE	CODE VALUE
1	DIELECTRIC CONSTANT	1.00+000 (NONE)	1.00+000 (NONE)
2	THICKNESS	1.00+003 METERS	1.00+002 MESH
3	CONDUCTIVITY	-1.00+000 MHO/M	-1.00+000 MHO/M
4	ATOMIC NUMBER	4.70+001 (NONE)	4.70+001 (NONE)
5	DELTA MAX >COEFF	1.00+000 (NONE)	3.09+000 (NONE)
6	E-MAX >DEPTH**1	8.00+001 KEV	1.58+002 ANG-01
7	RANGE	8.45+001 ANG.	6.93+001 ANG.
8	EXPONENT > RANGE	9.20+001 (NONE)	1.38+002 ANG.
9	RANGE > EXPONENT	7.94+001 ANG.	8.20+001 (NONE)
10	EXPONENT	1.74+000 (NONE)	1.74+000 (NONE)
11	YIELD FOR 1KEV PROTONS	4.90+001 (NONE)	4.90+001 (NONE)
12	MAX DE/DX FOR PROTONS	1.23+002 KEV	1.23+002 KEV
13	PHOTOCURRENT	2.90+005 A/M**2	2.90+005 A/M**2
14	SURFACE RESISTIVITY	-1.00+000 OHMS	-8.85+013 V-S/O
15	SPACE DISCHARGE POT'L	1.00+004 VOLTS	1.00+004 VOLTS
16	INTERNAL DISCHARGE POT'L	2.00+003 VOLTS	2.00+003 VOLTS
17	RADN INDUCEDCOND*YCOEFF	1.00+013 MHOMS3	1.00+013 MHOMS3
18	RADN INDUCEDCOND*YPOWER	1.00+000 (NONE)	1.00+000 (NONE)
19	DENSITY	1.00+003 KG/M**3	1.00+003 KG/M**3
20		2.00+001	-1.00+000

If a material called 'GOLD' or 'GOLDPD' or 'GOLDXXXX' is defined later with different properties the number of materials POLAR thinks it has will be increased by one, but the new properties will be effectively ignored. Multiple definition of materials should be avoided. Note, however, that if any of the default materials are explicitly defined before they are referred to in building block definitions then POLAR will make no attempt to find them in the list of default materials and the materials will not be multiple defined.

6.13 THE OBJECT DEFINITION FILE - ANOTHER EXAMPLE

We are now ready to bring together Sections 6.10-6.12 and examine the structure of the object definition file. The general form is shown in Figure 6/17. The materials are defined first, followed by the building blocks associated with each separate conductor. The use of COMMENT cards allow the logic of the definition of a complex object to be followed more easily. Finally the whole file is terminated with an 'ENDSAT' statement. An actual example is shown in Figure 6/18. It consists of a central RECTANGular body connected to two QSPHERES at the ends.

A 3D-VIEW (6.20) of the object produced by VEHICL is shown in Figure 6/19.

```

1:COMMENT WORKED EXAMPLE (SECTION 6.13)
2:COMMENT MATERIAL DEFINITIONS
3:COMMENT FOR PRESENT PURPOSES, ALL PROPERTIES ARE 'KAPTON'
4:COMMENT EXCEPT THICKER
5:TKAP
6: 3.5,.01,1.E-16,5.,2.1,.15,71.48,.60,
7: 312.1,1.77,.455,140.,.00002,1.E+16,1.E+4,2.E+3,
8: 1.E-13,1.,1.E+3,20.
9:COMMENT USE DEFAULT KAPTON AND GOLD VALUES
10:COMMENT DEFINE THE MAIN BODY AND TOP SPHERE TO BE ON
11:CONDUCTOR 1
12:COMMENT MAIN CUBOID BODY
13:RECTAN
14:CORNER -1 -1 -2
15:DELTAS 3 3 4
16:SURFACE +X GOLD
17:SURFACE -X KAPTON
18:SURFACE -Y KAPTON
19:SURFACE +Y KAPTON
20:SURFACE -Z KAPTON
21:SURFACE +Z TKAP
22:ENDOBJ
23:COMMENT TKAP SPHERE ON TOP
24:OSPHERE
25:CENTER 0 0 3
26:DIAMETER 3
27:SIDE 1
28:MATERIAL TKAP
29:ENDOBJ
30:COMMENT PUT THE BOTTOM SPHERE ON CONDUCTOR 2
31:CONDUCTOR 2
32:OSPHERE
33:CENTER 0 0 -4
34:DIAMETER 3
35:SIDE 1
36:MATERIAL GOLD
37:ENDOBJ
38:ENDSAT
EOF:38
0:

```

Figure 6/18. Object definition file.

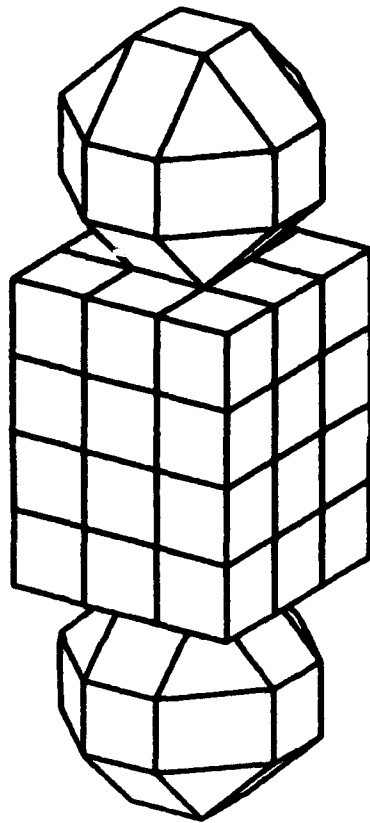


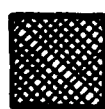
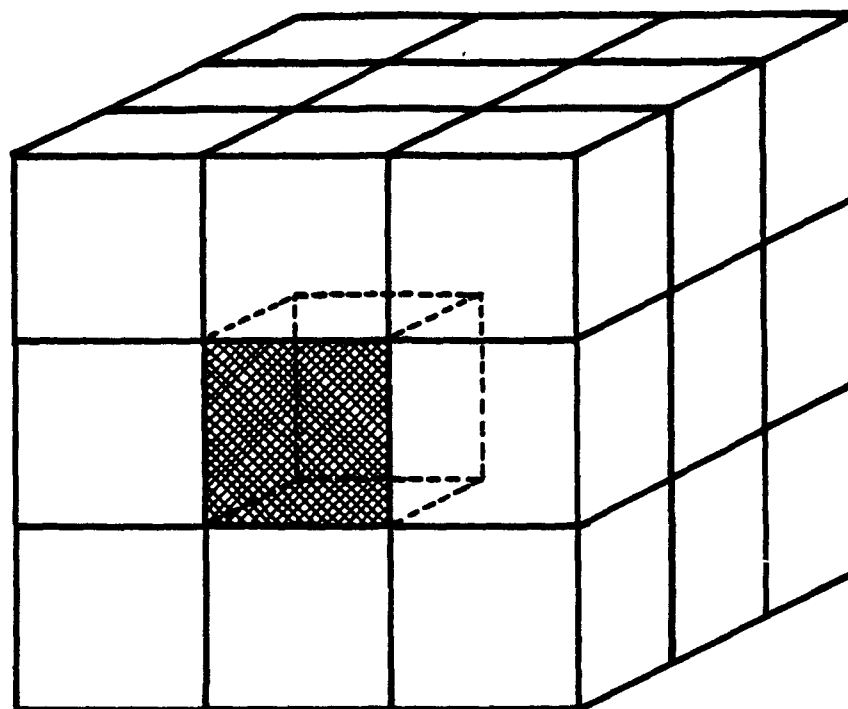
Figure 6/19. 3-D view of object produced by HIDCEL (hidden lines).

6.14 OBJECTS WITHIN OBJECTS: VARIEGATED SURFACES

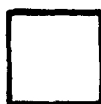
POLAR makes it easy to define surfaces that are made up of more than one material. For example, we may want to define one face of a cube to be mainly KAPTON but with a patch of say GOLD in the center (Figure 6/20). We begin by defining the cube with a KAPTON face. The center surface cell is then replaced with GOLD by defining a second smaller cube inside the first cube. The second cube is defined so that one of its faces is coincident with the KAPTON face. The surface common to both cubes is then associated with the material on the face of the second cube, which in this case is GOLD. This is shown in Figure 6/20.

The object definition file associated with this object has the form:

COMMENT	VARIEGATED CUBE
RECTAN	
CORNER	-2 -2 -2
DELTAS	3 3 3
SURFACE	+x KAPTON
SURFACE	-x KAPTON
SURFACE	+y KAPTON
SURFACE	-y KAPTON
SURFACE	+z KAPTON
SURFACE	-z KAPTON
ENDOBJ	
RECTAN	
CORNER	-1 -1 -1
DELTA	1 1 1
SURFACE	-z GOLD
ENDOBJ	
ENDSAT	



GOLD



KAPTON

Figure 6/20. A variegated surface definition.

The same principle can be applied to any of the building blocks. Exposed surface cells common to two or more building blocks are assigned to the material of the most recently defined block.

Two special building blocks are supplied specifically to create variegated surfaces. PATCHR and PATCHW define a RECTAN (cuboid) and a WEDGE respectively, that may be used to "patch" other objects without adding to POLAR's list of filled space. The use of actual RECTAN and WEDGE blocks inside others is also perfectly legitimate, but adds to the internally used surface list. The use of PATCHR and PATCHW reduces the likelihood of a problem occurring due to the list overflowing.

The object shown in Figure 6/20 could also be defined using PATCHR:

COMMENT	VARIEGATED CUBE (PATCHR)
RECTAN	
CORNER	-2 -2 -2
DELTAS	3 3 3
SURFACE	+X KAPTON
SURFACE	-X KAPTON
SURFACE	+Y KAPTON
SURFACE	-Y KAPTON
SURFACE	+Z KAPTON
SURFACE	-Z KAPTON
ENDOBJ	
PATCHR	
CORNER	-1 -1 -1
DELTAS	1 1 1
SURFACE	-Z GOLD
ENDOBJ	
ENDSAT	

6.20 VEHICL

The VEHICL module is used to interpret the object definition file (Section 6.10) in order to create the various tables and lists necessary for the other modules (Sections 5.23-5.25). It also can produce two separate kinds of object plots. By using appropriate keywords, either material or perspective plots are produced after the object file passes through the initial definition processing.

When using keyword inputs, each line (or card) is expected to contain one keyword followed by its list of parameters. After the keyword and parameters have been completely defined, the rest of the line is ignored. Several characters are ignored by the input routines, namely extra blanks between words (though some type of delimiter is necessary), equal signs (=), and commas (,). All input is read using free formatting with lower case characters being converted to upper case, and real numbers may be entered as integers.

VEHICL will need the following permanent files: 2. (for graphics output), 11. (MSIO), 19. (MSIO), and 20. (the object definition file, used as input). The following temporary scratch files will also be required: 3., 14., 17., 18., 21., and 27.. All of these files should be assigned with large storage limits. With the exception of file 20 (fort.20), VEHICL will define the files by itself.

6.21 VEHICL KEYWORDS

The specialized VEHICL keywords fall into three general categories; object definition, graphical output, and diagnostic output control. (See Table 6.21/1 for a brief summary of the keywords.) The diagnostic keywords are described in detail in Section 6.22. Additionally, all of the general POLAR keywords (Section 6.70), except SELECT, are recognized by the input routines. SELECT cannot be used by VEHICL because the grid information needed by the subroutine, MRBUF (5.30), is not necessarily well defined.

If an unrecognized keyword or an invalid use of a keyword is discovered, VEHICL issues a warning or an error message then terminates batch runs or reads the next input in the case of interactive runs. By default, VEHICL believes it is being run interactively. The keyword BATCH (see Section 6.70) will place VEHICL in its batch input mode.

The recognized keywords which control the object and grid definition are:

NXYZ

NXYZ defines the size of the object grid. This keyword should be defined for each execution of VEHICL since the default value for the grid dimensions may not be reasonable. An example of the use of the keyword is

NXYZ 6 7 8

This defines an object grid which has six nodes in the X-direction, seven nodes along the Y axis, and eight on the Z edge of the grid. Note that these numbers are in nodes, not elements. So a 1 x 1 x 1 cube would require a grid of at least 4 x 4 x 4 since object itself is 2 x 2 x 2 nodes and none of the object's vertices are allowed to touch the grid boundary. In general, extra space around the object is a good idea, especially if the object is centered asymmetrically in the grid and the object may be reoriented within the object grid later using ORIENT (6.30). These problems can also be avoided if an odd number of nodes along each axis are used. For more information concerning the declaration of object grid size, please see Section 6.10. The default is

NXYZ 17 17 33

DXMESH

DXMESH defines grid size in meters. For example,

DXMESH = 2.0

would define a grid spacing of two meters, while

DXMESH .01

sets the spacing to one centimeter. The default value is one meter. This size may be easily changed later during NTERAK.

OBJDEF

The OBJDEF keyword is used to change the unit number of the file containing the object definition file. For example,

OBJDEF 99

would instruct VEHICL to use file 99 as the object definition file. The default is to use file 20.

The standard use of this keyword is OBJDEF 5. In this case, the object definition should follow the VEHICL keyword input in the standard input (fortran file 5) file.

PREFIX

PREFIX is used to declare the object file name for the UNIVAC version of POLAR. The CYBER version expects the object definition to be located in file 20, so the PREFIX keyword is not needed. On UNIX machines, PREFIX is not needed and the object definition should be in fort.20. The UNIX default is "\$\$\$\$".

In the UNIVAC version, VEHICL must know the file's name in order to be able to find the input. If this keyword is omitted in the UNIVAC version, VEHICL will terminate with an error message. As an example,

PREFIX MICRO

causes VEHICL to read from the file MICROOBJ. Note the suffix OBJ must be used in all object file names. There is no default value for this keyword on the UNIVAC.

GRAPHICAL OUTPUT CONTROL KEYWORDS**MATPLOTS**

The keyword MATPLOTS controls the plotting of material plots. Material plots consist of six views of the object from each direction of all three axes. The surfaces of the satellite are filled in with different patterns depending on their material type. It should be noted only VEHICL can produce material plots, SHONTL is not able to duplicate them, although it can draw the same views without surface material patterns.

To control this feature, use

MATPLOTS YES

to turn it on and

MATPLOTS NO

to turn it off. By default, no material plots will be created.

The graphical output created by VEHICL is written on file 2, and needs to be interpreted by the post-processor graphics package (see Section 6.6). On the UNIVAC, this can be done automatically by using the PLOTDEST keyword described below.

MAKEPLOT

MAKEPLOT controls the creation of perspective plots. Perspective plots are views of the object as seen from infinity along a user defined vector. Both hidden line and transparent object drawings are produced. These views can be drawn by VEHICL; SHONTL is also able to generate them, though the syntax and keywords are different.

The keyword MAKEPLOT tells VEHICL how many views to expect. The actual viewing directions are defined using the PLOTDIR keyword described below. To set the number of views, use

MAKEPLOT N

where N is an integer from 0 to 8. A default set of views (two "random" vectors, from (2,3,5) and (-2,-3,-5)) can be requested using

MAKEPLOT DEFAULT

or just

MAKEPLOT

In both cases, PLOTDIR does not need to be used to define the views.

If no perspective plots are desired,

MAKEPLOT 0

will disable this feature. VEHICL by default will not make perspective plots.

The graphical output created by this command will be saved on file 2 and can be interpreted by the graphics post-processor described in Section 6.6. On the UNIVAC this may be done automatically by using the PLOTDEST command described below.

PLOTDIR

PLOTDIR is used to define viewing vectors for perspective plots. Each use of PLOTDIR describes one of the views from infinity requested by the MAKEPLOT keyword described above. To actually produce a plot, MAKEPLOT must be used. An example of the use of PLOTDIR is

PLOTDIR -2. +1.5 -1.

would be a view along $-2.0\hat{i} + 1.5\hat{j} - 1.0\hat{k}$ from infinity. Default directions can be defined with the MAKEPLOT keyword.

PLOTDEST

The keyword PLOTDEST is used to draw the plots created using the MATPLOTS or MAKEPLOT keywords described above immediately after VEHICL completes its execution. The keyword is only functional in the UNIVAC version of POLAR. Of course, it is always possible to draw the plots using the graphics post-processor (Section 6.6).

In the S-CUBED UNIVAC version of POLAR, several plotting devices are available. They are the electrostatic plotter, the Calcomp and the Tektronix 4014. To automatically draw plots on the later device, one must run VEHICL from the Tektronix where the plots are desired. The general form of the PLOTDEST command is

PLOTDEST destination

where destination can be blank, NONE, CALC (for Calcomp), ELEC (for electrostatic) and TEK (for Tektronix 4014). Leaving the destination blank or using NONE results in no plots being drawn at the conclusion of VEHICL (again the post-processor still can be used); this is the default condition.

OTHER VEHICL KEYWORDS

The following keywords are useful, or necessary, when running VEHICL. They are also described in Section 6.7.

BATCH

This keyword causes the input to be read in a batch mode. The main effect will be felt when erroneous input is discovered. If this occurs while in the batch mode, VEHICL will abort with an appropriate message. The default input mode for VEHICL is interactive (see description of INTERACT below). An example is

BATCH

which places the input routines in their batch input modes.

COMMENT

See description of REMARK below. The two keywords are equivalent.

DEFAULT

DEFAULT sets the VEHICL default values. In general, the defaults are for no diagnostic output, no plots, a grid spacing of 1 meter, and an interactive input mode. The keyword DEFAULTS is equivalent to DEFAULT. An example is

DEFAULT

VEHICL automatically calls the default routine before soliciting input. This keyword is most useful when using the interactive mode and the previous input has not been satisfactory, or in error.

END

The keyword END is used to signify the end of input to VEHICL. This keyword should always be used at the end of a runstream, but if an EOF (end-of-file) is encountered instead, VEHICL will still function normally.

An example is

END

No more input will be read at this point and VEHICL will begin operation.

INTERACT

The INTERACT keyword is used to place the VEHICL input routines in an interactive mode. This means that any errors encountered in the input runstream will generate an appropriate error or warning but will not cause VEHICL to terminate its execution. This is the default mode of input for VEHICL. To abort on the discovery of bad input, use the BATCH command described above. An example of the use of the keyword is

INTERACT

which will place VEHICL in an interactive input mode.

REMARK

This keyword is used to insert comments in a runstream. When a REMARK is encountered, the remainder of the input card will be ignored and a new card will be read. Any number of REMARKs may be used. An example of the use of the REMARK keyword is

REMARK THIS IS A REMARK

All of the data on the card following the first REMARK will be ignored. The keyword COMMENT (mentioned earlier) can be substituted for REMARK and is completely equivalent, for example,

COMMENT THIS IS A REMARK TOO.

And again, everything on the card which follows COMMENT will be ignored.

WHAT

This keyword prints out the current settings of the VEHICL options.

CIJ, RIJ

These two keywords define interconductor capacitances and resistances respectively. They should appear in the keyword input stream, not the object definition file. For definitions and usage instructions see Section 6.42.40.

TABLE 6.21/1
SUMMARY OF THE VEHICL KEYWORDS

BATCH	Places VEHICL input routines in a batch input mode. The default mode is interactive. (See INTERACT.)
COMMENT text	Causes text to be ignored by the input routines. Text may be any set of characters and numbers.
DEFAULT DEFAULTS	Resets options to default values. Called automatically at beginning of VEHICL.
DXMESH length	Defines grid spacing. Length is the grid spacing in meters. Default is 1 meter.
END	Makes the end of the VEHICL input. Should be included at the end of all runstreams.
INTERACT	Places VEHICL input routines in their interactive input modes. This is the default mode. (See also BATCH.)
MAKEPLOT option	Controls number of perspective plots. Valid options are DEFAULT (produces 2 default viewing directions) or an integer from 0 to 8. The default option is 0.
MATPLOTS option	Control material plot production. Valid options are YES and NO. YES produces 6 views of the object form *x, *y, *z directions. The default is NO.
NXYZ ix iy iz	Defines object grid size, where ix, iy, and iz are integers defining the number of nodes in the x, y and z directions, respectively. This keyword must be included in all VEHICL runstreams.
OBJDEF iunit	Take Object Definition from file iunit.
PREFIX name	Defines the file name containing the object definition. If the file name is CUBE OBJ, name would be replaced by CUBE. (Keyword applies to UNIVAC only.)
PLOTDEST option	Controls where plots are drawn at end of a VEHICL run (UNIVAC only). Valid options are blanks, NONE, CALC, ELEC, and TEXT. The default is NONE.
PLOTDIR x y z	Describes viewing direction, from infinity, used to draw a perspective plot.
REMARK text	Causes text to be ignored by the input routines. Text may be any set of characters and numbers.
WHAT	Display current VEHICL option settings.

6.22 VEHICL DIAGNOSTIC KEYWORDS

There are several levels of diagnostic output available from VEHICL by keyword instructions. None of it is of interest to the casual user and is mainly a remnant of the code development process. But sometimes errors, code modifications or just idle curiosity will require some of VEHICL's diagnostic output. By default, all VEHICL output flags will be turned off or set to the lowest possible values.

The following is a description of the appropriate diagnostic flags and their settings.

DIAG General VEHICL Output

- =0 No output.
- =1 A few crucial tidbits from the construction of the A-2, KSURF, LCEL, and LTBL lists.
- =2 More details concerning the construction of the various lists. KSURF in an unpacked format.
- =3 Still more information including octal lists.
- =4 Provides a great amount of details concerning VEHICL's actions.

IDIAGS(1) DCVCEL Information

- =0, 1, 2, 3 No output.
- =4 Output from DCVCEL during the creation of the SREL list. Also, information from the various VCUBE routines.
- =5 Additional DCVCEL data.

IDIAGS(2) CBUF Data Management

- =0, 1 Nothing.
- =2 Output from BUFSET.
- =3 Output from PAGER.
- =4 All of the information from PAGER and GRIDIO.

JDIAGS(1) SREL Information

- =0, 1 No output.
- =2 Print out LCEL and SREL lists. Also information from RECELL, SREL index to LCEL.
- =3 Print out preliminary LCEL list.
- =4 Detailed output from routines constructing LCEL and SREL lists.

JDIAGS(5) Double Point Data

- =0, 1, 2 Nothing.
- =3 Double point locations from SPDPNT.
- =4 More double point information from SPDPNT.

JDIAGS(9) MTLGEN Information

- =4 No information.
- =4 Material properties and final PSGM matrix.

KDIAGS(1) MSIO information (UNIVAC only)

- =1 MSIO routines echo calls to themselves.
- =2 Perform tracebacks when called.

KDIAGS(8) Edge list generation

- =0 None.
- =1 Entry and timing information.
- =2 Final results.
- =3 Intermediate data.

OBJPRT level Output from initial object definition routines.
Acceptable level keywords are NONE, SOME or ALL.

HIDPRT level Output from hidden line routine, HIDCEL. Valid level
keywords are YES and NO. Note that YES will generate
a vast amount of output.

IOGRID level Grid information. Valid level keywords are YES and
NO. Currently, this keyword is deactivated.

Some examples are

```
DIAG 3
IDIAGS(2) = 4
JDIAGS(9) 3
OBJPRT SOME
HIDPRT NO
```

This set of keywords would produce a high level of general information concerning the mechanics of the object definition (DIAG and OBJPRT), no MTLGEN or HIDCEL data (JDIAGS(9) and HIDPRT), and all of the available CBUF information (IDIAGS(2)).

6.23 AN EXAMPLE OF A VEHICL RUN

In Section 6.13, an object was defined (Figure 6.1/18) and a 3-D view of it was presented (Figure 6.1/19). Assuming the object definition kept in a file called XMPLOBJ (UNIVAC), or on file 20 (CYBER), and the necessary files have been declared large enough (6.20), Figure 6.2/1 shows the runstream (UNIVAC) used to draw Figure 6.1/19. The view used in the figure was the second one defined,

```
PLOTDIR 1.5 1. .5
```

```
1: @XGT VEHICL
2: BATCH
3: MAKEPLOT 2
4: PLOTDIR 1.5 1. .8
5: PLOTDIR 1.5 1. .5
6: REMARK PLOTDIR 3. 1. 2.
7: REMARK PLOTDIR 3. 5. 2.
8: REMARK PLOTDIR 2. 1. -3.
9: REMARK PLOTDIR 2. 3. -1.
10: PLOTDEST NONE
11: MATPLOTS NO
12: DIAG 1
13: JDIAGS(1) 2
14: JDIAGS(9) 4
15: REMARK IDIAGS(1) 2
16: NXVZ 16 16 16
17: DXMESH 1.
18: PREFIX XMPL
19: END
EOF:19
@>
```

Figure 6.2/1. VEHICL runstream used to draw Figure 6.1/19.

In addition, several diagnostic flags were used which produced about 2000 lines of unnecessary output.

The grid was chosen to comfortably hold the object. If this object had been defined to be run later using NTERAK, the grid would have been defined with more care with respect to the final desired size. It is best to keep from defining a great deal of extra space in the z-direction (or in the direction which becomes the z-direction after reorientation) to prevent the needless IO of oversized object grid size tables (Section 5.3).

6.24 TROUBLESHOOTING VEHICL

Usually VEHICL will provide an explanatory error message before it dies. The most common fatal error which currently has no message or at least an opaque, user hostile message, is from DCVCEL in SUREAL. The message, "#####-FATAL-FROM DCVCEL - ELEMENT NUMBER ... DOES NOT CORRESPOND TO LIST ENTRY", results from the grid being defined too small or the object being too close to one side.

What has happened is that somewhere the object has touched the edge of the object grid. The cure is simply to try again with a larger object grid.

Poorly defined objects or ambiguous double points also create difficult problems. These errors are typically called SCCYC errors after the routine which discovers them. Some useful advice can be found in Section 6.11. If the definition process has proceeded far enough, it may be possible to produce material or perspective plots as a visual aide (Sections 6.20 and 6.21). The SHONTL module may also be used to draw the object or to print out some of the list output which has been saved in the MSI0 files using CBUF storage functions (5.30).

Another restriction of object definition is that an element must be definable as one type. This mainly affects definitions where empty elements have more than one face with a diagonal on it. They are also known as type 2 elements (4.21.25) and come from the right triangle faces of wedges, tetrahedrons, and truncated cubes and the edges of slanted plates. Only one of these may touch the edge of an element.

6.30 ORIENT

In order that the object may be defined in an arbitrary direction or defined once and used in several orientations, it is necessary to be able to automatically reorient the satellite. The ORIENT module does this by rotating both the object and the object grid. This is necessary since NTERAK assumes the largest component of flow vector will be in the positive Z-direction, forcing a preferred direction on the problem. This module allows the same object to be studied in any flowing plasma without redefinition. Of course, if no rotation is necessary, ORIENT does not need to be used.

After ORIENT rotates the object, the original orientation and coordinate system is replaced by the new one. So keywords entered later which require a reference to spatial locations or directions should use the current reference frame.

ORIENT uses as input files the MSIO output files, 11 and 19, created by VEHICL. These files are also the output files, so they should be copied if the VEHICL output is to be preserved.

6.31 ORIENT KEYWORDS

In general, the keywords which were accepted by VEHICL (6.2) can be used in ORIENT since the two modules share many routines. Instead of redefining VEHICL keywords again, a reference to the section explaining the command will be given. The reader is also referred to the general keyword input section, 6.7, for more information. Table 6.31/1 contains a brief summary of ORIENT commands for convenience. The following is a description of the set of ORIENT keywords.

VMACH

VMACH is the plasma flow direction normalized to the ion acoustic speed. The entire problem will be rotated so that the largest component

of VMACH will be in the positive z-direction. The rotated VMACH will be stored and will be the default value for the NTERAK module. But the Mach vector can be changed later to any value so long as the largest component is still in the positive direction. An example of the command is

VMACH 0. 0. -8.

This would cause the object to be inverted so that the old negative z-direction becomes the positive z-direction. There is no default value for this command. If it is not defined ORIENT terminates with an error message. (This implies a default value of (0., 0., 1.), no rotation.)

For a description of the following ORIENT keywords see Table 6.31/1.

TABLE 6.31/1
A SUMMARY OF THE ORIENT KEYWORDS

BATCH	Places ORIENT input routines in a batch input mode. The default mode is interactive. (See also INTERACT.)
COMMENT text	Causes text to be ignored by the input routines. Text may be any set of characters and numbers.
END	Marks the end of the ORIENT input. Should be included at the end of all runstreams.
EXPAND x y z	Expand the grid x, y, and z dimensions
INTERACT	Places ORIENT input routines in their interactive input modes. This is the default mode. (See also BATCH.)
REMARK text	Causes text to be ignored by the input routines. Text may be any set of characters and numbers.
SHIFT x y z	Shift the object in x, y, and z direction
VMACH x y z	Defines the desired orientation by the sign and location of largest absolute component. x y z is the three vector describing the Mach vector normalized to the ion acoustic speed. This keyword must be defined for ORIENT to execute.

· WHAT

Display current settings of ORIENT options.

Diagnostic output from ORIENT is obtainable by using the VEHICL diagnostic flags defined by Section 6.22.

6.32 RUNNING ORIENT

After the VEHICL output files (or a copy of them) have been assigned, the runstream shown in Figure 6.32/1 could be used to rotate the satellite defined by Figure 6.1/18 and pictured in Figure 6.1/19 so that neither of the two spheres point in the ram direction. After execution, the reoriented object can be viewed using the SHONTL module to see where in the grid the object ended up.

```

OXQT ORIENT
BATCH
REMARK ROTATE FIGURE 6.1/19 TO BE
REMARK SIDEWAYS
VMACH -1. 0. 0.
REMARK THAT WAS EASYO
END

```

Figure 6.32/1 Sample ORIENT runstream.

Because the exact center of the grid is not necessarily on a node, the object may move a grid space in one (or more) directions. This can cause ORIENT to find an error during the SREL list creating. The error is due to the object touching the object grid boundary at some point. As previously mentioned (6.24), this error is cured by expanding the object grid. Unfortunately, only VEHICL can do this. So object must first be redefined in a larger grid by VEHICL, then rotated by VEHICL. It is a good practice to add an extra node on all sides of the grid if reasonable when an object is to be rotated.

An even better solution is to use an odd number of nodes along each axis. Then the center of the grid will fall on a node and the object should not touch the grid boundary in ORIENT if it did not in VEHICL.

Sometimes, it is not possible to rotate the satellite to the proper orientation in one call to ORIENT. For example, for plotting purposes one end of an asymmetric object needs to be in the +y direction while another needs to be in the -z direction. In these circumstances it is best to call ORIENT several times in sequence to rotate the object to the correct orientation in the grid.

6.40 NTERAK

The NTERAK module models the object-plasma interactions. Using the vehicle definition produced by VEHICL or ORIENT, NTERAK defines the plasma environment, the initial spacecraft potential, and then calculates the interaction of the plasma and the object using three main groups of subroutines.

The space potentials are calculated by PWASON (Sections 3.2, 4.2 to 4.4, and 5.5), and its associated routines, using the object surface voltages. These space potentials are then used to push particles to calculate new space charge densities for the attracted species. The ion currents to the object are also found in CURREN by pushing particles from a sheath boundary to the object (Sections 3.42, 3.6, 4.4, 4.53, and 5.6). These ion currents are combined with analytically calculated electron fluxes (Section 3.40 and 3.41) to electrically charge the vehicle in the set of subroutines led by CHARGE (Sections 3.5, 4.5, and 5.7). The calculation results can be printed as they are found. Or SHONTL can be used to print any of the final vectors or tables when the NTERAK run is complete.

The calculations and output are controlled using keyword input. Some notes were made concerning the use of the keyword input section in Section 6.20 and are worth repeating now.

When using keyword inputs, each line (or card) is expected to contain one keyword followed by its list of parameters. After the keyword and the anticipated parameters have been read, the rest of the line is ignored (allowing for personalized comments). Several characters are ignored by the input routines, namely extra blanks between words (though some type of delimiter is necessary), equal signs (=), and commas (,). All input is read using free formatting with lower case characters being converted to upper case, and real numbers may be entered as integers.

NTERAK requires MSIO Files 11 and 19 as input. Files 9, 10, 23, and 24 will all be used as temporary files (see Section 5.3). File 16 is used to save charging information and can be postprocessed by TRMTLK. The STATUS.JCO file is opened using unit number 8. All of these files should be assigned with large storage limits, especially File 10 as it will be used to manage the (trajectory tracking) particle lists. The UNIX version of POLAR expects files to be named fort 11, fort.19, etc. but will define the temporary files on its own.

6.41 NTERAK CONTROL KEYWORDS

In general the NTERAK keywords are similar to those recognized by VEHICL and ORIENT. Differences arise because NTERAK acts before reading the entire runstream. Input is solicited and interpreted until an instruction calling for calculation is encountered. When the computation is completed, input is again read until the next calculation command. This cycle repeats until the runstream has been exhausted. It is important that all input parameters be read before a calculation keyword is encountered, otherwise the calculation will be performed using possible inappropriate defaults. Moreover, the first encounter with one of the calculation keywords, PWASON, CURREN, or CHARGE, will initiate an evaluation of initial conditions. Thus, the bulk of grid and environmental parameters should be set before the first calculation keyword is encountered.

The following describes the calculation commands as well as general input control keywords. Several of these keywords have been defined elsewhere (Section 6.2) and are repeated here for convenience. The entire set of NTERAK keywords have been summarized in Section 6.45.

BATCH

This keyword causes the input to be read in batch mode. The main effect will be felt when erroneous input is discovered. If this occurs while in the batch mode, NTERAK will abort with an appropriate message.

The default input mode for NTERAK is interactive (see description of INTERACT below). An example is

BATCH

which places the input routines in their batch input modes.

CHARGE

This is a calculation keyword invokes the CHARGE subsection of NTERAK. If this is the first time a calculation keyword has been named, a full initialization will occur. The CHARGE subsection of NTERAK calculates surface charging of the object using analytic electron currents and either the ion currents found by CURREN or the random ion current to the object, depending on the history and characteristics of the problem. If CURREN has not been run, the random ion currents will be used (equivalent to a precharge). If it has, then the appropriate current will be used depending on the actions of the IONCUR routine (See Section 5.71).

CHARGE will also attempt to model the ion current as a function of voltage in order to stretch the usefulness of the CURREN results. At this point, it is recommended that only one timestep per CHARGE interaction be performed in most problems because of the difficulties of modeling the voltage dependence of the attracted, pushed species. For a description of the keywords which apply directly to the surface charging subsection of NTERAK, see Section 6.43.30. An example of the use of this keyword is:

CHARGE

This command would invoke the CHARGE subsection.

COMMENT

See the description of the keyword REMARK (below). The keywords are equivalent.

CURREN

The appearance of CURREN activates the particle pusher subsection of NTERAK. This section requires the presence of space potentials in order to function. Although the potentials are initialized to zero on the recognition of the first command keyword, it is strongly recommended that each use of CURREN be preceded at some point by a use of PWASON (see the following description of PWASON).

An example of the use of this keyword is:

CURREN

This invokes the particle pusher. Keywords which effect the control of the CURREN subsection of NTERAK are described in Section 6.43.20.

DEFAULT

DEFAULT resets the default NTERAK keyword values. In general, NTERAK defaults to an oxygen charging environment with no magnetic field. This environment is described in detail in Section 6.42.10. Reasonable amounts of useful output will be printed automatically. The default condition of the ISTART (defined below) is CONT. This was done to protect restart runs and their associated data from re-initialization. Please see the various keywords or the summary in Section 6.45 for individual keyword values.

NTERAK automatically sets the default constants at the start of a new problem (i.e., - the use of NTERAK after VEHICL or ORIENT). To reset the default values at any point, simply enter

DEFAULT

This will reset the default values. The keyword DEFAULTS is equivalent to DEFAULT and may be used if desired.

Since the values of all user input variables are saved at the end of each NTERAK run, it is best to use the DEFAULT keyword only on the initial run. Otherwise, all of the variable values different from the defaults will need to be defined again.

DONE

DONE is an internal keyword used by NTERAK to signal the last keyword of a run. It is equivalent to the keyword ENDRUN, discussed below. ENDRUN is the recommended keyword for this function. The two keywords can be used interchangeably at this point and for a more detailed description of its use and effects see ENDRUN.

END

This keyword was traditionally used to mark the end of the general starting or restarting of conditions of an NTERAK run. The input routines actually just ignore this keyword and will probably continue to do so in the future. An example is

END

which would be ignored by the NTERAK input routines.

ENDRUN

ENDRUN marks the end of an NTERAK run, and causes a variety of file indices and contours to be written out. If none of the other calculation commands (CHARGE, CURREN, and PWASON) have been called and this is the first use of NTERAK on VEHICL or ORIENT output files, then the initial conditions of the problem are set and saved before finishing the NTERAK run. If a calculation command was called or this is a continued run, NTERAK will end without additional calculations. For example,

PWASON

ENDRUN

after the PWASON subsection was completed, the ENDRUN keyword would be read and NTERAK would end normally, without an error.

HELP

HELP keyword features a separate set of "HELP Commands" to display different groups of NTERAK keyword settings and brief descriptions. For detailed keyword descriptions, the use of the manual is strongly recommended. Once entering HELP, only HELP keywords are recognized and the user must use the keyword QUIT to get back to normal NTERAK input.

Available HELP keywords are:

CHARGE	CHARGE keywords and values
CURREN	CURREN keywords and values
DIAG	Brief description of diagnostic flags
ENVIRON	Environment keywords and values
GI	Neutral density keywords and values
GRID	Grid information keywords and values
HELP	This HELP menu
PWASON	PWASON keywords and values
QUIT	Exit from HELP
WHAT	Overall NTERAK settings and diagnostics

IGICAL

IGICAL controls the initialization of the ion densities. By default, new neutral ion densities are calculated every time ISTART is set to NEW and then used in place of any existing ion densities. It is best to only calculate the ion densities once during the first NTERAK run since the process of calculating the geometric or neutral ion densities is fairly time consuming and costly. The recommended uses of this keyword are:

IGICAL YES

the first time NTERAK is executed and

IGICAL OLD

for all the following executions of NTERAK when ISTART is CONT. The first command calculates neutral ion densities and the second uses the previously calculated neutral ion densities plus any modifications which have occurred during the execution of the CURREN subsection. There other very useful options available for IGICAL, as well as other keywords affecting the ion density calculation, which are described in Section 6.42.30.

INTERACT

The INTERACT keyword is used to place the NTERAK input routines in an interactive mode. This means that any errors encountered in the input runstream will generate an appropriate error or warning but will not cause NTERAK to terminate its execution. This is the default mode of input for NTERAK. To cause an abort when bad input is discovered, use the BATCH command described above. An example of the use of the keyword is

INTERACT

which will place NTERAK in an interactive input mode. The interactive mode of input will also issue a prompt for the next input card.

ISTART

ISTART controls the initialization of the surface and space potentials and the ambient ion currents. The first time NTERAK is used after VEHICL or ORIENT or whenever a problem is to be reinitialized and started over, the following should be used:

ISTART NEW

This will reset potentials and define the random ion currents. Ion densities can also be reinitialized or reset depending on the setting of IGICAL (described above and in Section 6.42.30).

If a previous NTERAK run is being continued, use

ISTART CONT

This will prevent the resetting of previously calculated potentials and densities. CONT is used as the default ISTART value in order to protect previously calculated data. It should be noted the keyword controlling the calculation of the neutral ion densities, IGICAL, is defaulted so that when ISTART is set to NEW, ion densities are calculated. If new ion densities are not desired, IGICAL should be reset to use the appropriate densities.

LOOP

The LOOP keyword provides a convenient method to iterate on a fixed set of calculation modules. The format of the keyword command is:

```
LOOP number_iterations module_1 module_2 .... module_13
```

Where number_iterations is an integer defining the number of times the listed sequence of modules is to be executed. Up to thirteen modules may be listed. The acceptable module names are CHARGE, CURREN, and PWASON. These modules may be named in any order and may be repeated.

Different types of problems use the modules in different orders. For a floating potential, charging calculation the following LOOP command may be used to perform six iterations:

```
LOOP 6 PWASON CURREN CHARGE
```

While a fixed potential calculation, to find the plasma current collected by a spacecraft at a specific potential configuration, might use PWASON CURREN module list. Orbit limited charging calculations can be done with PWASON CHARGE.

Note that running NTERAK with a large number of iterations may take a while. If a job is happily running for an extended period of time and one wants to stop it without destroying the data files, the STOPRUN program can be used. This program is described in the utilities section 5.15.

PWASON

The keyword PWASON is used to activate the space potential solver. If this is the first NTERAK subsection to be executed (and it normally is), the initial values and constants are set, according to values of ISTART and IGICAL (see above). To invoke the Poisson solver, enter

```
PWASON
```

Then the object surface voltages, ion and electron densities, and boundary conditions will be used to find the space potentials. The boundary conditions used by POLAR are that the nodes just outside the defined grid (the virtual nodes) are at zero volts. This definition of the boundary needs to be taken into account especially when solving Laplacian or low density problems. The keywords affecting the PWASON subsection are described in Section 6.43.10.

REMARK

This keyword is used to insert comments into a runstream. When a REMARK is encountered, the remainder of the input card will be ignored and a new card will be read. Any number of REMARKs may be used. An example of the use of the REMARK keyword is:

```
REMARK    THIS IS A REMARK
```

All of data on the card following the first REMARK will be ignored. The keyword COMMENT (described earlier) can be substituted for REMARK and is completely equivalent, for example:

```
COMMENT   THIS IS A REMARK TOO.
```

And again, everything on card which follows COMMENT is be ignored.

SAVETEMP

This keyword is used to define the file save status for the temporary data files. Fortran files 9 and 10 contain intermediate data which may be useful for post-run analysis.

```
SAVETEMP ON
```

would cause the files 9 and 10 to not be deleted after use. The default is to delete them (SAVETEMP OFF) since they can be quite large.

WHAT

This keyword is used to display the current settings of various NTERAK keywords. WHAT produces output identical to HELP WHAT (See above).

6.42 KEYWORDS TO SET UP NTERAK

New NTERAK runs using object definitions from VEHICL or ORIENT need to define the environment and computational grid of the object. The plasma environment NTERAK uses by default may not be desirable. Keywords controlling these parameters are described in Sections 6.42.10 to 6.42.30. Some parts of the object definition also need defining (namely the initial state of the electrical model) and the keywords initializing these characteristics are explained in Section 6.42.40. The calculation grid may need expanding and Section 6.42.50 defines the pertinent keywords. Particle beams may be defined using the keywords described in Section 6.42.60.

Once these keywords have been defined, all of the POLAR modules will remember their values. Of course any keyword can be changed in later NTERAK runs to simulate changing environments or some other variable parameter. This feature is true for the keywords recognized by NTERAK including those in the PWASON, CURREN, and CHARGE subsections.

6.42.10 PLASMA ENVIRONMENT

Currently NTERAK allows the definition of two species of ions (one must be hydrogen) and hot and cold electron spectrums, as well as a photon environment.

ION AND COLD ELECTRON ENVIRONMENT

The ion and cold electron populations are assumed to be described by the same Maxwellian distribution.

DENS

DENS (or DEN1, they are equivalent) defines the density of the cold electrons and the combined density of the two ion species.

DENS 1.0E10

The above example would set DENS equal to 10^{10} meter⁻³. The input is expected in units of meters⁻³.

If a Laplacian problem is desired, the following command may be used:

DEN1 = 1.E-10

In this case DENS would be set to 10^{-10} m⁻³. Setting the density equal to zero is not recommended. (Note that the equal sign above is ignored by the input routine and is completely optional for all keywords.)

The default for DENS is 10^{10} m⁻³.

TEMP

The use of TEMP (or the equivalent keyword, TEMP1) defines the temperature both of the cold electron and ion populations.

TEMP 1.0

declares the temperature to be 1 eV. The temperature is assumed to be input in eV. The default temperature for the old species is 0.2 eV.

ESECNRGY

The average electron secondary energy, independent of source and material, is set using this keyword. For example,

ESECNRGY 2.0

would set the average electron secondary energy to +2.0 volts. This is also the default value.

AMUION

AMUION is used to input the ion mass in the units of atomic mass units. For example,

AMUION 14

would be used if nitrogen were the ionic species. The default value for AMUION is 16 (oxygen).

RATIH

RATIH is the ratio of the ion to hydrogen number densities. A large value of RATIH such as

RATIH 1.E20

would mean there are 1.0E20 as many ions of mass AMUION as there are of hydrogen. If the portion of ions or hydrogen drops below a minimum value (see RATMIN below), then the ion density is defined to be of only one species. The default value of RATIH is 1.0E20.

RATMIN

To change the value used to set the ion density to a single species, use the RATMIN keyword.

RATMIN 1.e-10

This would set RATMIN to its default value.

HOT ELECTRON ENVIRONMENT

The high energy electron environment encountered in polar orbits is represented in NTERAK by

$$\Phi(E) = AE^{-\alpha} + Cn_2 \frac{E}{(kT)^{3/2}} e^{-E/kT_2} + EB e^{-[(E - E_0)/\delta]^2} \quad (3.41-a)$$

(see Section 3.41), where $C = (2\pi^3 m_e)^{-1/2}$ and A , α , n_2 , T_2 , B , E_0 , and δ are parameters determined by the spectrum shape. The SHONTL module can be used to both define and draw the hot electron

spectrum. But when NTERAK is run after saving the spectrum parameters from SHONTL, the keyword DEFAULT should not be used. This is because the energetic electrons will be redefined.

The following keywords define the hot environment:

POWCO value	Power law coefficient (A in Eq. (3.41-a) above). Value is in units of $(m^2 \cdot sec \cdot str \cdot eV)^{-1}$ and the default is 1.4×10^2 .
PALPHA value	Power law exponent (α in Eq. (3.41-a)). Value is unitless and defaults to 1.2.
PCUTL value	Power law integration low end cutoff. Value is in eV. The default value is 50 eV.
PCUTH value	Power law integration high end cutoff. Value is in units of eV. Defaults to 1.6×10^6 .
DEN2 value	Energetic Maxwellian density (n_2 in Eq. (3.41-a)). Value is in units of m^{-3} and defaults to $4.2 \times 10^6 m^{-3}$.
TEMP2 value	Energetic Maxwellian temperature (T_2 in Eq. (3.41-a)). Value is in units of eV and defaults to 4.3×10^3 eV.

GAUCO value	Gaussian coefficient (B in Eq. (3.41-a)). Value is in units of $(\text{m}^2.\text{sec}.\text{str}.\text{eV})^{-1}$. Defaults to 8.8×10^5 .
ENAUT value	Gaussian peak (E_0 in Eq. (3.41-a)). Value in eV. Defaults to 8.2×10^3 eV.
DELTA value	e-folding width of Gaussian (δ in Eq. (3.41-a)). Defaults to 1.8×10^3 eV..

As an example, the default POLAR spectrum which approximates a spectrum observed by the DMSP satellite (Hardy, D. H., "The Worst Case Charging Environment," in Proceedings of the AFGL Workshop on Natural Charging of Large Space Structures in Near Earth Polar Orbit, 14-15 September 1982, AFGL-TR-83-0046, January 1983) would be defined by the following set of keywords:

DEN2	4.2E6
TEMP2	4.3E3
POWCO	1.4E12
PALPHA	1.2
GAUCO	8.8E5
ENAUT	8.2E3
DELTA	1.8E3
PCUTL	50.
PCUTH	1.E6

Figure 6.4/1 shows the fit of the POLAR spectrum to the experimental data.

DIFFERENTIAL ELECTRON FLUX

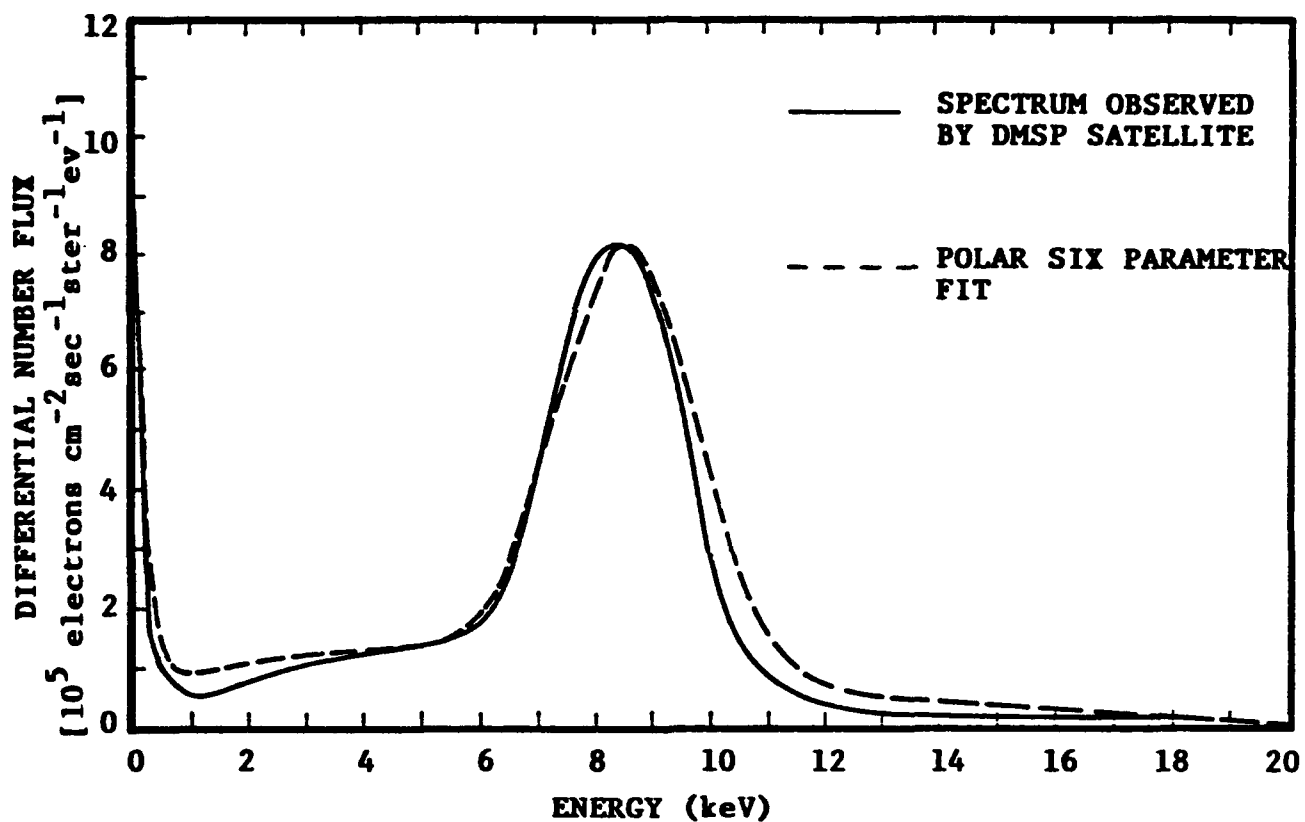


Figure 6.4/1. Comparison of the POLAR spectrum and the DMSP data (see reference in text).

PHOTON ENVIRONMENT

Photoemission and photoconductivity effects are modeled by POLAR. To control the current contributions from photo effects, the following keywords can be used to describe the light source:

SUNDIR

The card:

SUNDIR x y z

sets the direction from the spacecraft toward the sun. The vector is normalized by the code so the magnitude is not relevant. For example,

SUNDIR 2.0 2.0 0.

sets the direction of the sun between the positive X and Y axes on the XY plane. The default value is 0,1,0.

SUNINT

The card:

SUNINT intens

sets the sun intensity as a fraction or multiple of the natural sun intensity one earth distance from the sun. For any earth orbit exposed to the sun this should be 1.0, since orbit altitudes are negligible compared with the distance from the earth to the sun. Sun intensities differing from 1.0 (and 0.0) are used mainly for simulations of test tank environments using artificial UV sources, or for interplanetary spacecraft. For example

SUNINT 0.6

sets the sun intensity to 0.6 times its natural earth value.

The default is 0.0; in other words, the sun is "turned off" and the object is in shadow. The SUNINT keyword is used by NTERAK to control the presence of photo effects. None of the coding relating to photoemission are executed when the sun intensity is zero.

CONVEX

The card:

CONVEX yes_or_no

controls the calculation of surface cell shadowing effects. The keyword input

CONVEX YES

tells the shadowing algorithms the object is convex and none of the object surfaces are shadowed by any of the other surfaces. In this case, the intensity of photoradiation goes as the dot (inner) product of the surface normal and the sun direction. Surfaces determined to face away from the sun are considered totally shadowed.

The input

CONVEX NO

is the default value and directs the code to calculate surface-surface shadowing for all surfaces.

6.42.20 MAGNETIC FIELDS

The magnetic field environment of POLAR is controlled by three keywords. They are

BDIR B_x B_y B_z (B_x , B_y , B_z) define the direction of the magnetic field. This vector will be normalized by the input routines. Default is (-1, 0, 0).

BFIELD option This is a flag which turns the magnetic field on and off. Valid options are ON and OFF. The default is OFF.

BMAG value Magnitude of the magnetic field. Value is in gauss. Default is 0.4 gauss.

An example of the definition of a magnetic field is

BFIELD ON

BDIR 1 1.0 0

BMAG 0.25

This would define and turn on a constant magnetic field of

$$\frac{\sqrt{2}}{8} \hat{i} + \frac{\sqrt{2}}{8} \hat{j} \text{ gauss.}$$

The order of the three keywords is not important. The default field is $0.4 \hat{i}$ gauss.

6.42.30 NEUTRAL ION DENSITY

The neutral ion density calculation is controlled by a number of keywords. Since the calculation requires a rather slow computation for every element (cell) in the problem, several time saving options are available for special applications of NTERAK. There are two methods for the computation of neutral densities, the NEUDEN and the SHADO modules. No matter which module is used to calculate the initial densities, the SHADO module is used to calculate sheath wakes, if desired.

VMACH

The VMACH is used to define the plasma flow velocity Mach vector. For example,

```
VMACH 0. -3. 4.
```

would define a Mach vector of $-\hat{3} + \hat{4}$ Mach units. Or in MKS units, the flow velocity would be $\sqrt{kT/m_i} (-3\hat{j} + 4\hat{k})$ m/sec. Note that the z component of the vector must be positive and larger than the absolute values of both the x component and the y component.

To use NTERAK without a flowing plasma, use a small Mach vector, like

```
VMACH 0 0 .001
```

This is effectively a nonflowing plasma.

The default for VMACH is set by VEHICL ($8\hat{k}$) or by the Mach vector used in ORIENT to define the object rotation. (The Mach vector will have been rotated so that the largest absolute component is in the positive direction.)

IGICAL

IGICAL controls the calculation of the neutral densities by choosing between various alternate sets of ion densities. This keyword is ignored unless ISTART (Section 6.41) is set to NEW.

When IGICAL is not NEW, three basic types of options are available. They are to calculate neutral ion densities from geometric shadowing by the object, use previously calculated neutral ion densities, or to use final composite (sheath tracking plus neutral ions) from a previous calculation. In addition, an approximate electric field correction may be applied to a new calculation of neutral ion densities using the EFLDCOR keyword. It should be pointed out that the ion densities used by the code are normalized by the cold ion density as defined by DENS (6.42.10).

To save computation time with problems having low densities or no flow, one may want to set the combined ion density to 1.0 in each element. When the CURREN module is invoked, these densities are modified within the ion sheath to more accurate values. For example,

IGICAL NO

would cause a normalized density of 1.0 to be saved for each element in the problem.

A more commonly used option is to calculate the neutral ion densities using the flow vector (VMACH, described above) and several other keyword options defined below. For example,

IGICAL YES

would tell NTERAK to calculate neutral ion densities at each element. This is the default for new NTERAK runs (i.e., when ISTART=NEW, see Section 6.41).

IGICAL SHAD

is similar to IGICAL YES except the geometric shadowing method will be used to calculate neutral ion densities.

Previously calculated densities can be selected in three manners. These options presume the grid dimensions and Mach vectors have not been changed. To use composite (sheath and neutral) ion densities left in the files 11 and 19 by a previous run, one would enter

IGICAL OLDI

This prevents the ion densities from being changed at the start of a new or continued run. This is the default for continued runs (ISTART=CONT).

To restart a problem with previously calculated neutral ion densities, one would enter

IGICAL OLG I

The OLG I option causes the neutral ion densities to be copied over the ion densities used by PWASON.

The third way to utilize old density calculations should be used with care. If the following is used

IGICAL COPY

the neutral ion densities from a different set of data files will be copied over both the neutral ion density and composite ion densities used by NTERAK in the current files (11 and 19). The set of old densities are expected to be in files 12 and 20. File 12 is the file 11 produced by the previous run. File 20 corresponds to file 19. This feature is used to move ion densities for one object to another. The COPY option will work only if the objects are identical and the grids and environments to be used in both cases are the same. In other words, the only differences between the objects can be the surface material definitions. There is no error checking done when the command is executed and any errors caused by the misuse of COPY will turn up (if at all) in strange and mysterious manners. So be careful.

STHWAKE

To include the wake effect of the plasma sheath around the spacecraft, enter

STHWAKE ON

This will include the sheath wake effects at the end of a CURREN step. By default, this module is OFF.

EFLDCOR

The use of EFLDCOR will cause the neutral ion densities to be corrected analytically for electric field effects (see Section 5.6).

EFLDCOR YES

will turn on the electric field correction routines. By default, the electric field correction will be applied. To turn it off, enter

EFLDCOR NO

and no corrections will be made.

NPHI

NPHI controls the resolution of the zenith angle divisions for the neutral density calculation. Note that increasing the resolution also increases the calculation time.

NPHI 72

The above example would set the number of zenith angle divisions to 72. The default is 36 and is suitable for most applications. This keyword applies only to the NEUDEN module.

NTHETA

NTHETA controls the resolution in azimuthal directions during the neutral ion density calculation. Increasing the resolution will increase computational time proportionally.

NTHETA 180

would set the number of azimuthal angle divisions to 180. This is also the default value and has proven satisfactory for most applications. This keyword applies only to the NEUDEN module.

NADD

NADD is used to add extra vertices to the object shadow in velocity space during the neutral ion density calculation to increase resolution of the object. Care should be exercised when changing NADD since increasing NADD will slow down the calculation.

NADD 2

The above example would set NADD to 2. This is the default and should be appropriate for most situations. This keyword applies only to the NEUDEN module.

NHEXSH

NHEXSH controls the number of points used to resolve the object and the sheath during wake calculations by the SHADO module. To define the default, enter

NHEXSH 6000**NPHISH**

NPHISH sets the number of angular steps to be used to resolve the object and the sheath during wake calculations by the SHADO module.

NPHISH 16

This would set NPHISH to its default value.

SAVSET

Sometimes neutral ion densities may only be desired in subset of the entire grid. The SAVSET sets a flag indicating that only a subset of the ion densities needs to be calculated, and the rest of the elements may be set to 1.0. This keyword is only used by NTERAK when IGICAL=YES and ISTART=NEW (see above and Section 6.41).

For example,

SAVSET YES

would inform the geometric ion density calculator to calculate only a subset of values. The actual subset is defined using GISAVE (described below). The default for this keyword is NO, which implies that there are no subsets to be considered.

GISAVE

This keyword is only active when SAVSET YES is also included in the runstream (see above). GISAVE is used to define a subset of all the elements in the grid. The elements in the subset will have geometric ion densities calculated for them while those not in the set will be initialized to 1.0. The general form of the command is

GISAVE axis coordinate

where axis can be X, Y or Z and coordinate is a location on the axis in object coordinates and which is also inside the grid. In this manner, planes of elements may be defined up to a maximum of nine planes on each axis.

When multiple planes are defined, then the elements in each of the planes will have densities calculated for them. For example,

GISAVE X 2

GISAVE X 3

GISAVE Y 7

would cause the ion densities for the $x=2$, $x=3$ and $y=7$ planes to be calculated and all the elements not in these planes to be set to 1.0.

It should be noted that SHONTL requires two adjacent planes of densities in order to draw correct plots. If only one plane is calculated and drawn, the densities will be averaged with ones from the uncalculated plane. A maximum of eight planes on each axis may be defined.

TEMPRAT

During the neutral ion density calculation different ion and electron temperatures may be desired. For the ion density calculation only, the two can be defined to be different by using

TEMPRAT 10.

This defines the electron temperature to be ten times the ion temperature which is assumed to equal to TEMP (6.42.10). In other words, TEMPRAT is $T(\text{electron})/T(\text{ion})$ or $T(\text{electron})/\text{TEMP}$. The default temperature ratio is one, both temperatures are equal.

6.42.40 INITIAL VOLTAGES AND ELECTRIC MODEL

After VEHICL (and ORIENT), the electrical model has been completely defined for the insulators and their conductors, but the initial voltages and biases, and the conductor-to-conductor resistor and capacitor connections have not been set. These features are defined at the start of an NTERAK run or changed during a set of runs to simulate switching or other processes.

The following keywords are used to initialize and modify the spacecraft's electrical model:

CONDV

The CONDV keyword is used to set the conductor voltage. The card:

CONDV i v

sets the potential of conductor number i to v volts. All locations of the conductor are set, including both as an underlying conductor and as an exposed conductor. For example,

CONDV 2 -1000

sets conductor number 2 to -1000 volts.

The default value for conductors other than the ground conductor (conductor number 1) is the ground conductor's voltage, if there is no biasing (BIAS, see below); or the ground conductor's voltage plus the bias, if there is biasing. The ground conductor defaults to the value, $VLTFIX = -(kT/2e) \ln(m_i/m_e)$, where VLTFIX is an estimate of the simple thin sheath, no secondary electron, current equilibrium point. The value of VLTFIX may be changed by the user (see Section 6.43.30).

CONTCONDV i v

In continued NTERAK runs, CONTCONDV is used in place of CONDV to reset the potential of conductor number i to v volts. If omitted, values resulted from the previous NTERAK run will be used.

POTVAL

The POTVAL keyword is used to set the initial surface voltage of all the insulators. This keyword is used by NTERAK only when ISTART=NEW (Section 6.41). The card

POTVAL v

defines the initial insulator voltage to be v volts. The potential will be either the surface potential or the potential difference from the underlying conductor to the surface of the cell. This choice is controlled by the keyword INSULPOT (described below). Assuming POTVAL will define a surface potential,

POTVAL -100

sets the surface voltages of all of the insulating surfaces to -100 volts. The default value of POTVAL and INSULPOT are such that the insulator surface potentials are equal to the ground conductor voltage.

INSULPOT

INSULPOT is used to define the meaning of POTVAL (described above). If

INSULPOT DIFF

is used, POTVAL is the differential voltage from the underlying conductor to the surface of the insulating surface cell. If

INSULPOT CONS

is used instead, POTVAL is defined to be the surface voltage of the insulator surface cells. This is the default value for INSULPOT. For example,

INSULPOT DIFF

POTVAL -100

would set all the insulators to the ground conductor voltage less 100 volts. If

POTVAL -100

INSULPOT CONS

were used instead (the order of the keywords is not important), the insulator surface potentials would be equal to -100 volts.

INPOT

The INPOT keyword defines the method to be used to initialize surface potentials on the first NTERAK cycle. The form of the command is

INPOT flag

where the valid flags are CONS, PRE, and FLOAT. The default is CONS (constant). In this case the surface potential values are simply set using the other keywords and no special calculations are performed.

Sometimes when starting a new problem, it is desirable to use the random thermal ion currents to approximate the ion current and then calculate an initial set of potentials using the CHARGE module before the first call to PWASON (6.41). This extra step can save computation time by giving the Poisson solver a distribution of surface potentials which approximate the final voltages more accurately than constant potentials.

INPOT PRE

The above example would automatically call CHARGE at the start of a new run. This is equivalent to using the default, "INPOT CONS", and executing the CHARGE module first, since the ion currents are initially set to the thermal values in new runs.

The FLOA (float) option is used when the surface to surface secondary electron currents are expected to require the use of constant normal electric field boundary conditions in the spatial potential solver. Normally, PWASON and CHARGE are capable of choosing the appropriate boundary conditions, but on the first cycle it is best to set the flag in order to reduce wasted computation time.

Floating boundary conditions normally arise in situations where the photoemission currents dominate surface charging, though other environments exist which may generate large, low energy secondary currents.

FIXP

The card:

FIXP n v

fixes the potential of conductor number n to v volts. For example

FIXP 4 -6000

sets conductor 4 to a potential of -6000 volts, where it remains, fixed for all future potential calculations. The most common use of this option is to ground conductor 1:

FIXP 1 0.

The default behavior is for all conductors to float freely.

Note that fixing the ground conductor when there are biased conductors (see BIAS below) will have the side effect of fixing all of the biased conductors, but that it is not allowed to fix the ground conductor (conductor number 1) by fixing one of the biased conductors.

BIAS

The card:

BIAS i v

causes conductor i to be biased by v volts relative to conductor 1.

Conductor 1 is usually the spacecraft ground. For example, the card

BIAS 3 -1000

causes conductor 3 to always be 1000 volts more negative than conductor 1. If conductor 1 were floating or fixed at -300 V then conductor 3 would have a floating or fixed potential of -1300 V. The BIAS cards for each conductor must be entered in ascending order. Thus any card for conductor 3 will be rejected unless conductor 2 has been biased. Cards need only be included for those conductors that the user wants to be biased. The default behavior for a conductor not biased or fixed (see FIXP above) is to float independently.

FLOAT

The card:

FLOAT i

removes the effect of all previous BIAS's and FIXP's affecting the specified conductor number i. For example

FLOAT 4

allows a previously biased or fixed conductor 4 to float freely again. This is necessary because options FIXP and BIAS are remembered from previous runs. The card:

FLOAT

with no conductor number causes all previous FIXP and BIAS commands to be cancelled for all conductors; i.e., all conductors float freely.

CIJ

There are two sources of capacitance between conductors. The stray capacitances that are determined by the throughspace electric fields are not currently calculated by POLAR. The larger mechanical capacitances due to conductors being glued together or separated by dielectric films must be specified by the user. The card:

CIJ k l c

sets this "mechanical connection" capacitance between conductor number k and l to c farads. For example the card:

CIJ 2 3 1.E-8

sets the "mechanical" capacitance between conductor 2 and 3 to 1×10^{-8} farads. The default interconductor capacitance is zero, so failure to define this value may cause differential charging between the two conductors in question to occur unrealistically fast.

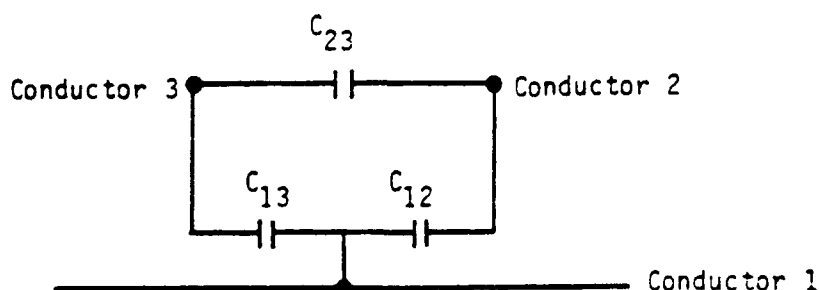
When multiple definitions of capacitances are made POLAR includes implicit as well as explicit connections in its calculation. For example

CIJ 1 2 1.E-10

CIJ 1 3 2.E-10

CIJ 2 3 3.E-11

defines explicit capacitive connections between conductors 1 and 2, 1 and 3 and 2 and 3. However, 1 is also connected to 3 via 2, and so on. The circuit diagram used by POLAR has the form:



RIJ

POLAR has the capability of treating explicitly specified conduction among the various conducting segments. Otherwise, no connection is assumed, and an infinite resistance is used as the default. Since infinity is difficult to represent on a computer, a zero value is used internally to flag infinity and resistances less than or equal to 1 ohm are not allowed. Interconductor resistances are specified in a similar manner to interconductor capacitances by the card

$$RIJ \quad i \quad j \quad r$$

where i, j are conductor indices, and r is the direct interconductor resistance in ohms. Resistances less than 1 ohm will be ignored and the infinity default will remain in effect. Use of very low resistivities to effectively short two conductors together is not recommended. The user should, instead, change one conductor to the other and rerun VEHICL.

6.42.50 GRID SIZE CONTROL

At the start of a new NTERAK run, the computational grid extensions need to be defined (Section 5.20). Only the object definition grid has been defined at this point and usually it is too small to perform a calculation.

There are two main features which control the grid size: the grid spacing and the number of grid points. The defaults for these features are the values left after completion of VEHICL or ORIENT, whichever was executed last.

DXMESH

The grid spacing is controlled by the DXMESH keyword. The general form is

DXMESH ℓ

where ℓ is the grid length in meters.

DXMESH .5

defines the grid spacing to half a meter. The default value for NTERAK is the value used by VEHICL or ORIENT. The default for ORIENT is the value from VEHICL and VEHICL's default is 1.0 meter.

NUMBER OF GRID POINTS

The number of grid points in a problem depends on several factors. (See Figure 5.20/1 and Section 5.20.) The grid will be at least as large as the object definition grid. If the Mach vector (6.42.30) is defined with non-zero x and y components, the grid will grow enough in the x and y directions to keep the object grid contained in the smallest stepped grid (NXGRTH in Figure 5.20/1).

The grid can further be expanded in any of the six axial directions ($+\hat{x}$, $+\hat{y}$, or $+\hat{z}$). The following keywords are used to extend the grid along either direction of the three axes:

NXADNB adds nodes in the -X direction
 NXADNT adds nodes in the +X direction
 NYADNB adds nodes in the -Y direction
 NYADNT adds nodes in the +Y direction
 NZADON adds nodes in the -Z direction
 NZTAIL adds nodes in the +Z direction

For example, the grid shown by the Figure 5.20/1 would be defined by

NXADNT 1
 NXADNB 1
 NZTAIL 7
 NZADON 2

The add-ons along the y axis have not been defined since they are not apparent from the figure.

The only limits on the grid size is that the product of number of the grid points on the x(NX), y(NY) and z(NZ) axis must satisfy

$$(NX+2)(NY+2) \leq 5000$$

$$(NY+2)(NZ+2) \leq 5000$$

$$(NX+2)(NZ+2) \leq 5000$$

in order to plot the calculation results and the following limits on the Z coordinates.

$$MINZ \leq -50$$

$$MAXZ \leq 100$$

$$MAXZ - MINZ + 1 \leq 100$$

where MINZ and MAXZ are, respectively, the minimum and maximum Z coordinates of the grid points in object grid coordinates (where the lowest (x,y,z) corner of the object definition grid is defined to be equal to (1,1,1)).

The size of the data buffer common block, /CBUF/, also limits the size of the grid but can be easily expanded until machine limits are reached. The buffer should be able to hold as much as $20 \times (NX) \times (NY)$ words of data.

6.42.60 PARTICLE BEAMS

The CHARGE module models the effects of particle beams on the charging of conductors. Currently, the space charge effects of beams as well as surface-beam interactions are not modeled. Trajectory plots of beams can be generated using SHONTL. Although not checked on input, it is best to define only one particle beam per conductor. When making plots, this restriction can be ignored. The following describes the keywords for defining beams and their default values. The words in upper case characters are keywords and those in lower case represent sets of keywords. The general forms of beam descriptors are

BEAM ALL on-off

This turns all of the described beams on or off. The keywords on-off can be replaced by ON or OFF. Or

on-off = {ON, OFF}

A single beam can be turned on or off with

BEAM n on-off

where n is an integer and is the beam number. A maximum of twenty beams may be defined and saved. If it has not been described, the default values will be used.

There are two methods which can be used to describe a particle beam. The first is to turn the beam on using one of the above commands. Then modify the default values individually until reaching the desired particle beam definition. The syntax used to modify one feature of a beam is

BEAM n beam-char char-value

where n is the beam number, beam-char is a beam characteristic, and char-value is the value of the beam characteristic. The definable beam characteristics are summarized in Table 6.42/1.

TABLE 6.42/1
BEAM CHARACTERISTICS
(a) Keyword Definition

beam-char (keyword)	Definition
LOCATION	spatial location of beam
DIRECTION	direction beam aimed
COND	number of conductor current
CURRENT	beam current
AREA	cross-sectional area of beam, assumed to be circular
ENERGY	beam energy
MASS	mass of beam particles
CHARGE	charge of beam particles

TABLE 6.42/1
BEAM CHARACTERISTICS
(b) Input Syntax

beam-char (keyword)	char-value (syntax)	char-value (format)	units	default value
LOCATION	x y z	3 real	grid coords	0 0 0
DIRECTION	x y z	3 real	N/A	1 1 1
COND	n	integer	none	1
CURRENT	value	real	amps	-.1
AREA	value	real	m ²	$4\pi(.01)^2$
ENERGY	value	real	KeV	1
MASS	value	real	AMU	m_e/m_H^*
CHARGE	value	real	electron charge	-1**

* default value 1 if CURRENT > 0. m_e is electron mass and m_H is proton mass.

** default value 1 if CURRENT > 0.

The second way to define a beam is enter all of the beam characteristics on a single input card. This may not be a practical method in some cases due to the 80 character limit of the input card. The first method also provides a form of self-documentation which is easier to read. The order of the beam characteristics is LOCATION, DIRECTION, COND, CURRENT, AREA, ENERGY, MASS and CHARGE where these keywords are defined in Table 6.42/1a. The syntax of the command is

```
BEAM  n ALL      loc-x, loc-y, loc-z, dir-x, dir-y, dir-z, cond,
                      current, area, energy, mass, charge
```

where loc stands for location and dir for direction. Values should be placed between commas. The commas may be replaced by spaces but values may be left out between commas if the default is correct. The default values describe a kilovolt electron beam drawing 100 mAmps from the ground conductor and having a 2 cm diameter. The location and direction of the default beam is arbitrary.

A 200 mAmp, 1 KeV hydrogen beam drawing current from conductor 2 is defined by

```
BEAM 1 ON
BEAM 1 COND 2
BEAM 1 CURRENT 200.
BEAM 1 CHARGE 1.
```

The second method could also be used. One way is

```
BEAM 1 ALL ,,,,,,2, 200.,,, 1.
```

An apparent extra comma is used after the ALL keyword since a delimiter is needed between ALL and the x coordinate of the beam location vector. The values between the commas are filled by the defaults summarized in Table 6.42/1b. Clearly, the first example is easier to read at a glance.

6.42.70 SHEATH IONIZATION

By checking the total ionization of neutral ions within the sheath, NTERAK is able to adjust the sheath radius to account for enhanced ion densities. It is also able to check to the case when sheath ionization begins to dominate the calculation. The sheath boundary becomes unstable and will expand quickly to fill the computational grid. When this occurs a message is printed and the run is killed. The neutrals are assumed to be defined by AMUION (6.42.10). The keywords necessary to turn on sheath ionization effects are:

SHEIONZ turns on/off sheath ionization effects. To turn on ionization effects

SHEIONZ ON

To turn them off

SHEIONZ OFF

By default, sheath ionization effects are off.

NEUTDEN This keyword sets the neutralized ion density. By default, the neutral density is $10^{12}/\text{m}^3$. For example,

NEUTDEN 1.E+14

would define the neutral density to be $10^{14}/\text{m}^3$.

IONZCROSS This defines a probability cross-section for the electron/neutral atom interaction. For example

IONZCROSS 1.e-20

would set the interaction cross-section to be 10^{-20} m^2 , which is also the default.

6.43 NTERAK SUBSECTION CONTROL

The calculations performed by NTERAK to model spacecraft charging can be conveniently separated into three major subsections. They are the Poisson potential solver PWASON, the particle pusher CURREN used to find ion currents and densities, and the electrical model of the spacecraft, CHARGE. The keywords which control these modules are described in the following three sections.

6.43.10 PWASON (POISSON POTENTIAL SOLVER)

The PWASON subsection of NTERAK is used to calculate space potentials given a set of surface voltages for an object (Sections 6.42.40, 5.70, 4.50), a computational grid (Sections 6.42.50, 5.20, 4.10), a set of ion densities (Sections 6.42.30, 5.60, 4.40), and a zero potential boundary condition (Sections 5.50, 4.20) PWASON involves a double layer of iteration. Innermost is a conjugate gradient potential solver (Section 4.31) that uses a linearized Boltzmann electron contribution to the space charge (Section 4.44). This is referred to as the Conjugate Gradient Iteration. This is, in turn, iterated upon to update the linearized Boltzmann electron space charge derivative (SCRN, Section 4.44). This outer iteration is known as the space charge iteration. Both the conjugate gradient and space charge iterations terminate on either a convergence test or an iteration limit. There is an analytic formula available to generate space charge densities instead of pushing particles.

PWASON

The keyword PWASON is used to activate the space potential solver (also see Section 6.41). For example.

PWASON

would invoke the PWASON subsection. All of the keywords which control the subsection must be defined before the appearance of PWASON in the runstream if the various defaults are not appropriate.

ENORMCHK

PWASON will recalculate the space potentials if a surface's boundary condition has changed as a result of the new space potentials. This, by default, only happens on the first call to PWASON when there has been no PRECHG step. If it is desirable to check and recalculate after every PWASON call, use

ENORMCHK YES

To turn this off, use

ENORMCHK NO

The latter example is the default condition.

This keyword is mainly intended for use with positive potentials when secondary electrons are influencing charging.

MAXITS

The MAXITS keyword defines the maximum number of space charge iterations (Section 4.44 and introduction above). For example

MAXITS 3

would set the number of iterations upon space charge to three, the default. A MAXITS of one should be used for low densities and/or short Debye lengths where the space charge will have little impact on the final solution. Higher numbers of iterations, twenty, may be necessary when factors of density, temperature, object potential, and object size conspire to make the repelled electron space charge contribution significant over large regions of the model. If analytic densities are being used, a larger value should be used.

MAXITC

The keyword MAXITC controls the maximum number of conjugate gradient iterations. If none of the convergence criteria have been reached after MAXITC steps, the potential data is saved and the NTERAK run is terminated with an error message. Saving the potentials allows the run to be restarted if desired.

To define a maximum appropriate for a low density or Laplacian problem, an input such as

MAXITC 30

would be appropriate. The default is 20 which is enough for most densities.

MINITC

The MINITC keyword is used to set the minimum number of conjugate gradient iterations to be performed (see also MAXITC above).

This essentially causes the convergence criteria to be ignored until the MINITC +1 iteration. To change the default, enter

MINITC 1

This forces at least one conjugate gradient calculation to be performed. The default value is 2 and will not need to be changed for most applications.

POTCON

The keyword POTCON is used to determine satisfactory conjugate gradient convergence. The code uses

$$\log_{10} \frac{RDOTR(1)}{RDOTR(\nu)}$$

to check convergence, where RDOTR is a measure of the accumulated error of the potential solver (Sections 4.32, 5.50) found at the first and at the current (ν) on conjugate gradient iteration. The number calculated above is compared to POTCON and if it is larger than POTCON, the conjugate gradient iteration terminates.

To set this variable, use a keyword instruction similar to
POTCON 4

This example would require approximately four orders of convergence during the conjugate gradient loop unless MAXITC (see above) is exceeded. The default value for POTCON is six which provides a good compromise between accurate potentials (e.e., more convergence) and approximately correct, as when approaching a total problem convergence, this parameter can be lowered to four or five.

RDRMIN

The keyword RDRMIN is an absolute measure of convergence of the space potentials (see Sections 4.31, 5.50). RDRMIN defines a value of RDOTR (a dimensionless measure of the accumulated error in the

potential solution) which is considered a completely converged solution. If RDOTR is less than RDRMIN, no further conjugate gradient iterations are required or performed. For example,

RDRMIN 1

would define a total error of 1 volt for the problem. The default is calculated by multiplying the number of nodes in the problem by 0.0001 volts. This is appropriate for most applications. For most problems, the conjugate gradient iteration should be terminated by POTCON (defined above). RDRMIN provides a lower limit to prevent the needless use of calculation time.

RMSCONV

The space charge iteration loop of PWASON can end by satisfying a RMS convergence condition. The value of the convergence level is entered using the RMSCONV keyword. For example,

RMSCONV .01

would cause the space charge loop to continue until the root-mean-square of the space potentials was less than or equal to .01 volt (or MAXITS was exceeded). The default is the maximum of TEMP (TEMP is described in 6.42.10) and the absolute value of STHPOT (see 6.43.20).

RMSOFF

The root-mean-squared convergence criteria in the space charge loop of PWASON can be turned on and off via this keyword.

RMSOFF OFF

turns off the convergence checking, while

RMSOFF ON

turns it on. By default, the convergence check will be made. The convergence check will never be made after the first space charge iteration, so at least two iterations will be made (unless MAXITS = 1). See also RMSCONV.

SQALPH

SQALPH is the space charge limiting factor. The interaction of this keyword and the space potentials is complex and the reader is referred to Section 4.44.2. Example:

SQALPH 4

would define SQALPH to be 4. The default is 2.667. and this value is appropriate to most charging cases. Larger values of SQALPH increase charge near the sheath edge, a value of 3 may help stabilize the sheath location if its position is oscillating. For situations where objects are discharging, a value of 2 is recommended.

MOTIONDEN

The MOTIONDEN keyword is used to turn on or off the analytic spatial densities during potential calculations.

MOTIONDEN ON

This would cause PWASON to ignore the neutral and pushed densities and use the analytic densities. The default is OFF.

MIXDEN

MIXDEN is used to play with the diagonalization of the finite element density information when pushed particle densities are being used. Its use is not recommended. The default is .5.

MIXDENMO 0.

MIXDENMO modifies the diagonalization of the finite element density information when the analytic densities are being used. Its use is not recommended. The default is 0.

PDIE

The PDIE keyword defines the maximum allowable positive voltage in the calculation (Sections 4.30, 4.44, 5.50). NTERAK originally had no model for positive space potentials and PDIE was used to detect this situation. The keyword remains for debugging purposes.

PDIE 50

The above example sets the PDIE to +50 volts. The default value is +9999 volts.

6.43.20 CURREN (ION CURRENT CALCULATION)

The CURREN subsection of NTERAK is used to calculate the ion densities inside of the sheath (Sections 3.60, 4.42, 5.60), and the ion currents to the surface of the spacecraft (Sections 5.71, 4.53) using outside-in particle pushing. The module obviously requires a set of space potentials in order to perform the ion calculations.

CURREN

The appearance of the keyword CURREN in the input runstream causes the CURREN subsection of NTERAK to be executed (see Sections 4.40, 4.53, 5.60). For example

CURREN

would invoke the ion particle pushing subsection (see also Section 6.41). Since this keyword activates the CURREN module upon its appearance, all keywords controlling the various features of the module need to be defined (if the defaults are not appropriate) before it is read as input.

IPCNT

IPCNT defines the maximum number of -Z (left) and +Z (right) particle pushing sequences allowed by CURREN. The particle pusher used by NTERAK moves each particle within a specific z-slice until that particle leaves the slice, the sheath, the grid, or hits the object. After all the particles in the slice have been pushed out, the particles in the next slice are moved. In this manner the entire grid is swept through, first in the +Z direction, then in the -Z direction. IPCNT is the maximum number of these right/left pushing sweeps and is necessary to prevent "stable" particle orbits from causing infinite loops.

IPCNT 2

The above example would limit pushing to two left and two right sweeps so trajectories would terminate after two or three turns in the z-direction, depending on the particle's original velocity direction. The default for IPCNT is 3.

IPQUIT

It is possible for stable, trapped orbits to occur during particle pushing. To stop trajectories orbiting in the Z plane, the number of particles still active after each right/left pass through the grid is compared to the previous pass. If the number of particles is the same after IPQUIT passes, pushing is halted. For example

IPQUIT 2

would stop pushing after two consecutive passes without a change in the number of active particles. The default is two.

XYLIMIT

Trapped orbits are also possible in the xy plane. The particles are limited to moving into a maximum of XYLIMIT elements without leaving the plane (a change in the integer portion of the Z coordinate).

XYLIMIT 100

would limit particles to entering 100 elements before marking the particle as lost and moving the next particle. The default is 40.

STEPLIM

When step pushing particles, the estimated time used to move may be too small (for example, due to magnetic fields) for the particle to cross the element in a reasonable number of steps. So after a certain number steps, specified using STEPLIM, the timestep is doubled. For example,

STEPLIM 10

would double the timestep after every ten steps in the same element.
The default is 20. (See also MAXSTPIN.)

MAXSTPIN

The number of timestep doublings caused by STEPLIM (above) is limited by MAXSTPIN if

MAXSTPIN 3

was used, on the fourth attempt to double the timestep the particle would be marked as lost and the next particle would be pushed. The default value is 5.

STHPOT

The keyword STHPOT defines the sheath edge potential where the initial position and velocity of the particles will be calculated. The flux represented by the initial particles is defined using CURPOT (see below). To define a sheath at -2.0 volts, enter

STHPOT -2.

The default sheath "mode" is

STHPOT PSIM

which causes the sheath edge potential to be the lesser (more negative) of CURPOT or ϕ_m kT/e, where

$$\phi_m = 2 n (SQALTP * (\lambda_D / DXMESH)^2).$$

(SEE 3.60, 4.42.1, and 4.44.2 for explanations.)

In general, care should be taken to insure that the sheath boundary defined by STHPOT is at least a mesh length away from the edge of the grid boundary. If the sheath comes too close to the grid boundary, the accuracy of the solution will be adversely affected.

CURPOT

The CURPOT keyword defines the presheath edge potential (3.60, 4.44.2) at which the presheath fluxes are calculated. CURPOT is also used as an upper limit for the default of STHPOT, described above. To redefine this variable, use input in the form of

CURPOT -.069

The above example defines the presheath edge potential to be -.069 volts. The default voltage is $-.45 \times \text{TEMP}$ (TEMP is described in Section 6.42.10).

NTABLE

NTABLE controls the number of table entries used by STHCAL (5.62.21) to calculate the presheath focusing weights.

NTABLE 20

The above example would call for 20 entries to be used for interpolating the presheath weights. The default is ten which has proven sufficient for most cases.

MAGSTH

The MAGSTH keyword should be used when collecting magnetically limited electron presheath fluxes.

MAGSTH ON

This turns magnetic flux tube restricted electron presheath currents ON. The default is OFF.

6.43.30 CHARGE (SURFACE CHARGER)

The CHARGE module is responsible for computing the response of the vehicle electrical model (Sections 3.50, 4.50) to the plasma. The ion and electron currents are either the ambient ion currents (Section 3.43), the currents calculated by CURREN (Section 6.43.20), or orbit limited currents. The energetic electron currents are calculated numerically in this module using a parametric representation of the electron spectrum (Section 3.41; keywords, Section 6.42.10).

CHARGE is able to calculate the change in surface potentials using orbit limited currents, space charge limited currents, or hybrid currents. The hybrid currents are surface currents where the total current to the object is determined by finding the current through the space charge limited sheath. The proportion of the current to a given surface is found by dividing the orbit limited current to the surface by the total orbit limited current to the object.

The charging algorithm (discussed in detail in Sections 5.73 and 4.50) is controlled by the following keywords. The more crucial keywords are those controlling the allowable voltage change, the timestep size used by the algorithm, and the ones which define the charging regime.

CHARGE

The keyword CHARGE is used to invoke the CHARGE module (5.73 and 4.50). For example

CHARGE

would cause the surface charging section of NTERAK to begin execution. The keywords which are used to control the electrical model should be defined (if the default values are not appropriate) before this keyword is entered. See Section 6.41 for a more detailed explanation of this keyword.

SPCLIM, ORBLIM, and ORBSPC

These keywords define the current module to use for the attracted species. The default method can be requested with

SPCLIM

The SPCLIM keyword is used for space charge limited (pushed particle) currents. ORBLIM will turn on the orbit limited current models. For intermediate charging environments, ORBSPC uses the space charge sheath current to renormalize the orbit limited surface currents.

DELTAT

The keyword DELTAT (Δt) defines the size of the timestep to be taken by the charging algorithm. Long timesteps (a large DELTAT) allow the algorithm to converge quickly on an equilibrium solution without regard to the actual charging time history that one would expect or perhaps measure during an experiment. Short timesteps will require many more iterations to yield information concerning long timescale charging effects, like differential charging. To define DELTAT, enter an instruction like

DELTAT .01

which would be a timestep of 10 msec. The units of the input are assumed to be seconds and the default is 1 second.

MAXITT

The MAXITT keyword defines the number of iterations to be performed by the CHARGE modules. Since there are currently no convergence criteria or any other checks for completed calculation, this keyword is the sole means of controlling CHARGE. To set the maximum number of timestep iterations for the charging algorithm, use input of the form

MAXITT 4

which would set the loop counter to four. The default value is 2.

For situations where current balance is expected to occur due to the pushed and secondary currents, a value of one for MAXITT should be used. This is due to the lack of a good model in the code of the voltage dependence of the pushed specie currents. In cases where current balance is achieved between analytically calculated incident currents and their secondaries, multiple iterations of the charging algorithm may be used.

DVLIM

The DVLIM keyword defines a limit on the change in voltage during a single timestep. There are two major circumstances where DVLIM limits the voltage step.

The first is during the first implicit stage when the current derivative is estimated. The voltage at which the current balances for a specific surface is estimated to be $XDVFAC \times DVLIM$ away from the original (at the start of the step) voltage. XDVFAC is described below and has a default value of two.

The second instance when DVLIM is used to limit the voltage change is when the sign of the total current to a surface changes and the surface voltage appears to be moving away from the current balance voltage limits defined by VLTFIX and VFIXHI (possibly due to the influence of other surfaces or the underlying conductor). In this case, the surface voltage is constrained in the second and final stage of a CHARGE iteration to diverge no more than DVLIM from its initial value at the start of the step. This is done to stabilize the problem.

An example of the use of the keyword is

DVLIM 100

which defines DVLIM to be 100 volts. The units of DVLIM are volts. The default value is 1000 volts.

XDVFAC

This keyword is used as a multiplying factor for DVLIM (described above) when calculating an estimate for the voltage change necessary to reach an equilibrium voltage during the first implicit stage of the iteration. For example

XDVFAC 2

would set XDVFAC to two, which is also the default.

If, for example, a satellite initially at -1.0 V was placed in a strong charging environment with DVLIM = 100 volts and XDVFAC = 2, the satellite would charge no more than about -200 V during a single, long timestep.

DVTEST

When a surface voltage is near to the voltage which will establish a current balance, the change in the surface's voltage during the previous iteration is used like DVLIM to limit the voltage change and to force convergence on a solution. Of course during the first iteration this variable is undefined. DVTEST is used instead. For example,

DVTEST 2.0

would limit the surface voltage change to 2.0 volts when the equilibrium point is passed (found by noticing a change in the sign of the total current). The default value is 5 volts.

DVMAX1

DVMAX1 v

is the absolute measure of convergence of surface charge. Charge iterations will be halted if every surface potential changed within v volts from the previous charge iteration. For example,

DVMAX1 1.

will cause charge iteration to continue until dV of every surface is \leq 1. volts (or MAXITT is exceeded or conditions of DVMAX2 is satisfied). Default value for DVMAX1 is 5.*temp.

DVMAX2

DVMAX2 $\frac{\#}{\#}$ is the relative measure of convergence of surface charge. Charge iterations will be halted when changes in surface voltages are within $\frac{\#}{\#}$ (percentage) of previous voltages. For example,

DVMAX2 .05

will stop the charge iteration when $dV/oldV \leq .05$ or 5% (oldV is previous iteration V). Of course, MAXITT and DVMAX1 still can cause the charge iteration to halt. Default for DVMAX2 is .05.

VWIGGL

When a surface voltage is at current equilibrium, it is convenient to keep the surface voltage charging by a small amount in order to avoid numerical difficulties. VWIGGL is used to introduce a small amount of noise to prevent these problems. For example,

VWIGGL .01

would force the voltage to change at least 0.01 volts each iteration. The default is 0.001 volts and is adequate for most applications.

VLTFIX

The keyword VLTFIX is used to estimate an upper bound on the surface voltage producing a current balance in noncharging cases. Currently, this is the value which will be used for all of the material types in the problem. For example,

VLTFIX -.01

defines the upper boundary to be -0.01 volt. The default is found using plasma parameters defined in Section 6.42.10 to solve

$$\text{VLTFIX} = - (\text{TEMP}/2) * \ln \left(\frac{\text{AMUION} * 67 * 10^{-27}}{9.1 * 10^{-31}} \right)$$

This default should be appropriate for most problems.

VFIXHI

The keyword VFIXHI is used as an estimate of the lower boundary of the equilibrium voltage or a charging surface. It is used to prevent destabilizing voltage swings when at the current balancing voltage. For example,

VFIXHI -8000

defines VFIXHI to be -8000.0 volts. The default is -10,000.0 volts which should be adequate for most cases.

FXFORM

The FXFORM keyword allows the code to choose the best matrix formulation of the charging equations (FXFORM = NO) or forces the CHARGE module to use the undiagonalized matrix formulation (FXFORM = YES) (see Section 5.73.2). In general, it is best to use the diagonalized form if possible since this form has smaller off-diagonal terms which enables the matrix solver to work more efficiently.

To allow the code to choose the best formulation based on the charging behavior of the problem, enter

FXFORM NO

To force the code not to diagonalize the charging equations, enter

FXFORM YES

The default is FXFORM set to NO and should not need changing for most applications.

USELIM

The USELIM keyword controls part of the voltage limiting process used to find a surface voltage to be used to find the current derivative term for the second stage implicit step (see Section 5.73). This keyword will almost never need to be used.

When a surface charges quickly towards one of the estimated current balance voltages defined by VFIXHI and VLTFIX (see above) at more than DVLIM (also described above) volts, this flag determines whether the voltage change should be limited to DVLIM or allowed to proceed more rapidly towards a solution.

To limit the charging process, enter

USELIM YES

and allow the code to charge rapidly under these special circumstances,
enter

USELIM NO

The default value is NO and should be used for most problems.

6.44 NTERAK OUTPUT CONTROL

Currently the bulk of the NTERAK keywords produce output that is oriented more towards code diagnostics than for studying calculation results. As the code develops, keywords are added to provide the user with increasingly easier methods of accessing calculations. Presently, the greatest difficulty facing a potential user is sifting through the output, or output control.

The first section defines keywords which reduce the output by printing useful subsets of the calculation results. The second section provides a description of all the available diagnostic controls and complete definitions and examples of keywords which control both calculation and diagnostic output. Some keywords appear in both sections to facilitate convenient referencing.

6.44.10 CALCULATION MONITORING

The output interface with the user has not been fully developed. But many features do exist to present calculational results that are adequate for most purposes. Any of the information stored using the MRBUF (Section 5.33), as well as graphical representations of data, are available via the SHONTL module (for SHONTL operating instructions see Section 6.50), for the last cycle of the calculation. If intermediate results are desired in graphical form, it is advisable to break a calculation into several steps and save these intermediate files (11 and 19).

NTERAK can also provide information during execution. Originally designed as diagnostic tools, several keywords can also be used to print results as they are computed. This section focuses upon these keywords.

OUTPUT

The OUTPUT keyword provides a convenient method of coarsely controlling the output from several major sections of NTERAK. The general form of the command is

OUTPUT subsection quantity

where quantity is the desired amount of output from subsection. Information from the subsections can be printed at three levels of verbosity: HIGH, LOW, and OFF. The sections of NTERAK which are controllable via the OUTPUT command are:

TIMER	-	code speed information
NEUDEN	-	neutral ion density calculation
PWASON	-	Poisson potential solver
CURREN	-	particle pushing module (ion currents and ion densities within the sheath)
CHARGE	-	electrical charging model

Some examples of the use of OUTPUT are:

OUTPUT TIMER HIGH

OUTPUT CHARGE OFF

OUTPUT PWASON LOW

The first example causes a continuous monitoring of the execution time at numerous points within the code, most notably after each mass I/O operation. The second example stops all of the output from the CHARGE subsection while the third example produces the minimal amount of information necessary to following the space potential calculation. The default quantity of output for all subsections is LOW. The SELECT (described below) can be used to remove or reduce the amount of information printed at HIGH levels. The OUTPUT keyword is discussed further in the next section, 6.44.20.

SELECT

The SELECT keyword can be used to choose a specific set of Z-slices from grid sized arrays. It can also be used to turn off (or on) the output of any of the data stored by MRBUF (5.30). This is useful when working with the output keywords set to high levels.

A variety of options are available with SELECT. The five general forms of the command are:

SELECT name ALL

SELECT name OFF

SELECT name NONE

SELECT name list ENDLIST

SELECT LIST list-number list ENDLIST

SELECT name LIST list-number

where name is the name used by the buffering system (5.30) to reference the data, list is a list of Z-slices using object coordinates, and list-number is an integer from 1 to 5.

The first three examples turn the output of a buffered data set on (ALL) or off (OFF and NONE). By default, all data sets will be printed if requested during execution (i.e., set to ALL). The keyword OFF resets the output selection to the default value. The NONE keyword causes no information to be printed when NTERAK requests that data set. Note that the SELECT keyword will not produce any output. It only controls the output which is produced by other keywords.

The fourth example of SELECT is the basic form used to print only certain specified slices of sliced, grid sized data sets like POT (space potentials) or DION (ion densities). When a set of slices for a number of data sets are desired, the last two examples can be combined to reduce the effort necessary to print them by defining a list, then selecting the list-number instead of retyping the list.

When printing out bit packed lists, SELECT can be used to control whether the data is printed in an unpacked or packed form.

```
SELECT NICE
will print the information in decoded format and
SELECT OCTL
```

or

```
SELECT PACK
will choose a packed format. The default is to print a packed format.
These keyword commands set a switch which affects all bit packed data
types, but currently, only the KSURFOP (surface information list)
recognizes the switch.
```

Examples:

```
SELECT TSRV OFF
SELECT SRFV ALL
OUTPUT CHARGE LOW
```

The above set of keywords would cause the surface voltage list (SRFV) to be printed at the end of each CHARGE iteration when the CHARGE module is executed. The trial surface voltage lists calculated during a CHARGE iteration would not be printed even though the third command (OUTPUT) would generally produce them.

More examples:

```
OUTPUT CURREN HIGH
SELECT DION 4 3 2 5 -1 0 ENDLIST
SELECT RHOI -1 0 2 3 4 5 ENDLIST
```

In this case, when CURREN is activated the same set of slices for both the composite ion densities (DION) and the ion densities found by particle pushing (RHOI) would be printed (among other things) by the OUTPUT command. The set of RHOI slices printed is $Z = -1, 0, 2, 3, 4, 5$. The slices are printed in the same order they are defined.

```
SELECT LIST 3 5 4 3 -1 0 2 ENDLIST
SELECT RHOI LIST 3
OUTPUT CURREN HIGH
SELECT DION LIST 3
```

The same set of slices as in the previous example (but in a different order) will be printed, and DION now uses the same list.

IGIOUT

The IGIOUT keyword is used to print the neutral ion densities at the start of a new or continued NTERAK run.

```
IGIOUT YES
```

The above example will cause the ion densities to be printed.

```
IGIOUT NO
```

The second example will prevent the production of the ion densities. By default, the ion densities are not printed at the start of each run. This output can be restricted with SELECT options.

ISPOUT

The ISPOUT keyword controls output of the grid sized arrays, POT, QUSD, and SCRN (Section 4.44.1), from the space charge calculation of PWASON, the space potential Poisson solver. The general form of the command is

ISPOUT option

where the options are

- | | | |
|-------|---|--|
| Full | - | print the arrays on each of the MAXITS
space charge iterations. |
| Final | - | print the array after the last space charge
iteration. |
| No | - | print nothing. |

For example,

ISPOUT FINAL

would print the final potentials at the end of use of PWASON. This is the default value. The output can be further restricted with SELECT options.

IOCONG

IOCONG controls output from the conjugate gradient portion of the Poisson potential solver, PWASON (Section 4.21.1 or 4.31). The general form of the command is

IOCONG option

where the valid options are FULL, PART, and NO. The FULL option prints detailed information from the conjugate gradient routine and is intended to be used mainly as a diagnostic output level. PART generates a smaller amount of information and NO prints none at all. For example,

IOCONG PART

is the command using the PART option. The NO option is the default value since this information should be of no interest to most users.

6.44.20 DIAGNOSTIC OUTPUT

During the development of POLAR, the various print statements used for debugging were implemented in such a way that they could be turned on and off via keywords. The capability of diagnostic output from most areas of NTERAK still remains and has proven useful on occasions when new bugs are discovered. Most of these debugging print statements are controlled by the "diag" flags summarized in Section 6.45.10 and which are discussed here. Other options have been added to improve the usefulness of the diagnostic flags (for example, SELECT, Section 4.51.10). Since part of the job of debugging involves checking the correctness of intermediate physical quantities needed to study spacecraft charging, many of the keywords described in this section and the previous section on calculation monitoring overlap.

The division between the two sections is intended to separate the more general use keywords from the more obscure output options. This section offers additional insight into the calculations performed by POLAR and NTERAK in particular.

OUTPUT

The OUTPUT keyword is used to interface a single command to a set of relevant diagnostic flags (described below). Section 6.44.10 discusses the use of OUTPUT in detail. Here the actual output produced by the command shall be defined.

As before, the general form of the command is

OUTPUT subsection quantity

Each subsection contains an associated set of diagnostic flags. Table 6.44/1 shows the settings of the flags defined by the OUTPUT keyword. The diagnostic flags and keywords in the table are all discussed later in this section. These flags can be set individually to customize the output (see below).

Table 6.44/1. Flag Settings for the Subsections by Quantity Level

<u>Subsection</u>	<u>Flags</u>	Flag Settings at Quantity		
		<u>OFF</u>	<u>LOW*</u>	<u>HIGH</u>
TIMER	IDIAGS (4)	0	2	3
NEUDEN	JDIAGS (4)	0	1	2
	KDIAGS (3)	0	0	2
PWASON	ISPOUT	NO	NO	FULL
	IDIAGS (1)	0	1	1
	KDIAGS (7)	0	1	2
CURREN	IDIAGS (6)	0	1	2
	IDIAGS (9)	0	1	2
	JDIAGS (3)	0	1	2
	JDIAGS (6)	0	0	2
	JDIAGS (9)	0	0	2
CHARGE	IDIAGS (8)	0	1	4
	JDIAGS (3)	0	1	2
	JDIAGS (8)	0	1	2

*The LOW setting is the default setting for all subsections.

SELECT

The SELECT keyword is used to choose a subset of the slices of the grid for output or to turn off the output of a data set altogether. Section 6.44.10 contains a complete discussion of the options and uses of the keyword.

IGIOUT

The IGIOUT keyword is used to print the ion densities at the start of a new or continued NTERAK run.

IGIOUT YES

The above example will cause the ion densities to be printed.

IGIOUT NO

The second example will prevent the production of the ion densities. By default, the ion densities are not printed at the start of each run.

ISPOUT

The ISPOUT keyword controls output from the space charge portion of the potential calculation (PWASON, see Section 4.44). The general form of the command is

ISPOUT option

where the valid options are FULL, PART, NO, FINAL and LAST. The FULL option causes the screening potentials (SCRN), the space charge used (QUSD), the space potentials (POT) and the surface voltages (SRFV, if they have changed during PWASON) at the conclusion of each space charge iteration. The PART option produces the same output as the FULL except for the omission of the screening potentials. The FINAL and LAST options are equivalent and print the same data sets as the FULL except only once at the conclusion of the space charge loop. The NO option generates no output. The default value for ISPOUT is NO.

The use of SELECT (see Section 6.44.10) is recommended when using this keyword as a great deal of information is produced.

IOCONG

The IOCONG keyword controls output from the conjugate gradient portion of the potential solver. There are two loops in PWASON, one upon the space charge and the other over the conjugate gradient. The conjugate gradient iteration is the inner loop. Therefore it is possible to generate huge masses of output using this keyword. In general, it should be of no interest to the user.

Nonetheless, here is a description of it and its options. The general form is

IOCONG options

where the valid options are FULL, PART, and NO. The FULL option prints all of the variables associated with the potential solution (see Section 4.21 for the variables used). The PART option prints potentials from various points during the calculation. The NO option is the default option and generates no output.

IOGRID

The staggered grid used by POLAR requires the definition of a coordinate reference point in each z-slice (see Section 5.2). Each of the different grids and data types (nodal or element centered values) need a different set of reference points. The IOGRID keyword prints the offsets used for each of the data types. To print the reference point offsets, enter

IOGRID YES

To not print them,

IOGRID NO

The latter is the default.

SAVETEMP

This keyword is used to save the temporary files 9 and 10 after an NTERAK run. SAVETEMP ON enables this feature; and SAVFTEMP OFF, the default, causes the temporary files to be deleted when the run finishes.

Note: RHOI's and RHOE's are stored in fort.9 and particle lists are stored in fort.10. So, SAVETEMP must be ON if these data are to be examined in SHONTL.

DIAGNOSTIC FLAGS

There are currently 25 NTERAK diagnostic flags. In this section only a terse definition of each flag is given. The specific output generated by each level of the flags can be found in Section 6.45.10. The general form of the command is

keyword level

where keyword is the diagnostic flag (see below) and level is the integer value used to control the amount of information produced. In general, larger levels produce larger amounts of output. The valid diagnostic keywords appear below.

IDIAGS(1)

Used by PWASON, this keyword provides conjugate gradient convergence information at low levels. As the level is increased, more detailed code mechanics information is printed.

IDIAGS(2)

This flag can be used to monitor the mechanics of the buffered data system (CBUF, Section 5.30). The lower diagnostic levels are concerned with which data types are currently in memory. The higher levels follow the input/output operations of the data.

IDIAGS(3)

This flag is used to check the actions of VERTIO, the subroutine which handles node data associated with specific elements. For example PWASON loops over VERTIO, using it to find and replace the potentials at the nodes of each element in the grid.

IDIAGS(4)

At low levels, this flag checks the time spent in each of the subsections of NTERAK. The level checks time spent in certain loops of the modules and at the highest level, input/output speeds are checked.

IDIAGS(5)

This keyword turns on a self-diagnostic package within NTERAK. Currently, the addressing system used to reference grid stored data and the core locations of certain addresses are checked.

IDIAGS(6)

Used by the CURREN module, this keyword is used to monitor the behavior of particles as they are pushed from the sheath to the object. At low levels, groups of particles are watched. The higher levels watch the individual particle movements and particle energy conservation.

IDIAGS(7)

This flag is used to receive information from the QSELT routine which calculates the charge per node and the screening factor for elements (a part of the PWASON module).

IDIAGS(8)

This flag controls the output from the CHARGE module. At lower diagnostic levels, only crucial sets of data are printed. While as the level increases, more and more of the arrays and lists used by the charging algorithm are produced. At the highest level the currents to individual surfaces are listed in detail.

IDIAGS(9)

This flag is used to monitor the calculation of ion currents using the output of the particle pusher. The lowest level prints the SRFI computed by IONCUR. As the level is raised, weighting calculation details then particle lists from CURREN are printed.

JDIAGS(2)

This flag is used to receive information from the presheath calculation.

JDIAGS(3)

Use this flag to receive information concerning the ambient ion currents calculated by AMBCUR.

JDIAGS(4)

This flag produces information about the neutral ion densities calculated by IONDEN. It is used to receive progress notes for speed checking and for calculated densities for filled and partially filled elements.

JDIAGS(5)

At low levels, the space potentials calculated for double points are printed using this flag. At higher levels, the input of the double point location list and the double point potentials to major modules of NTERAK can be monitored.

JDIAGS(6)

This flag causes the particle pushed ion densities to be printed at the conclusion of the CURREN module.

JDIAGS(7)

Sometimes during the code development of CURREN particles would be lost in the pushing mechanics. This flag controls the actions to be taken when particle pushing errors occur. At low levels, information is printed which indicates how a particle has been lost. Higher diagnostic levels will terminate NTERAK if a certain threshold percentage of particles are lost.

JDIAGS(8)

This flag provides an alternative to IDIAGS(8) for controlling output from the CHARGE module. This flag focuses mainly on the surface voltages and currents calculated by the subsection, not on the general algorithm as the other keyword does.

JDIAGS(9)

This flag is used to print the ion densities (DION) and real grid sized element table (LTYF) computed during particle pushing. These are the final results found after combining the initial densities with the newly calculated densities.

KDIAGS(1)

This keyword is used to locate and monitor the subroutine calls to the various mass storage input/output routines.

KDIAGS(3)

This flag is used to print the tables used to make the electric field correction on the neutral ion densities.

KDIAGS(4)

This flag is used to monitor the portion of the particle sheath definition call DBLCHK. This routine backtracks along particle trajectories to verify that the initial particle has not come from a high potential region or through part of the object.

KDIAGS(5)

Not widely implemented throughout the code, this keyword controls the diagnostics and actions which occur when an error condition is detected. At low levels, the offending variables and arguments are printed in addition to an error message. At higher levels, arrays, lists, etc., are printed and at the highest level (only on the UNIVAC presently) an interactive dump mode is entered.

KDIAGS(6)

This keyword is used to print diagnostic information pertinent to the shadowing calculation done for photosheath calculations. At low levels, rotation angles are printed. As the diagnostic level increases, surface shadow results and intermediate results are produced. At the highest levels, hidden line plots are generated from the sun direction (SUNDIR, 6.42.10) and stored on file 2 (fort.2 on UNIX versions of POLAR). These pictures can be plotted on a graphics terminal by the TEKLOT or JUPITER graphics drivers.

KDIAGS(7)

KDIAGS(7) controls the output relating to the normal electric fields on the object surfaces. When set to low levels, the normal electric fields are printed at the beginning and end of each PWASON cycle. As the diagnostic level increases, the surface voltage and IFLT list (5.33) is also printed, then at higher levels the three lists are printed more frequently during PWASON, after each space charge iteration. At the highest level of diagnostic output, this keyword prints the parallel electric fields for each space charge step.

KDIAGS(9)

This keyword is used to produce surface boundary condition data. PWASON and CHARGE communicate with each other via lists containing the boundary conditions for each surface. At low levels this flag prints information from the PWASON side of the interface and as the level of output is increased more information on the CHARGE perspective is printed.

LDIAGS(1)

This is used to produce intermediate data during shado sheath and shado GI calculations. At high settings, two or more, this keyword will produce lengthy and unfriendly output (which is useful for debugging purposes).

LDIAGS(2)

This keyword prints out particle trajectory information from PUSHER.

IBIAGS(1)

This flag controls diagnostics from CURREN concerning the magnetic field.

IBIAGS(2)

This flag controls magnetic field diagnostics from the CHARGE module.

6.45. SUMMARY OF NTERAK KEYWORDS

NTERAK is the program that calculates the plasma interaction and vehicle charging. Its keyword control language has both 'verbs' and 'noun'-options. A run-language sentence is constructed in the reverse-Polish sense, i.e., nouns first, verbs last. The NTERAK verbs are:

PWASON -	invokes the Poisson solution
CURREN -	invokes the sheath particle tracking
CHARGE -	invokes the circuit model update of surface potentials
ENDRUN -	run finished

The following is a list of the NTERAK keyword options. The first column is the keyword, the second column lists the possible responses. Literal options are listed with the default underlined. Numerical responses are indicated by the default value in parentheses followed by the variable type I, for integer; F, for floating point real number, E, for exponent type, i.e., 1.2E12, and L, for literal (no quotes).

<u>OPTION</u>	<u>VALUE, FORMAT</u>	<u>EXPLANATION</u>
<u>BATCH</u> <u>INTERACT</u>		Keyword only to select operation mode, turn prompts on or off, etc. non-critical.
<u>DEFAULTS</u>		Set or reset default option.
<u>ISTART</u>	<u>NEW</u>	New run. Allows for the calculation of new GI's or the use of old ones if the computation grids are identical. Space potentials will be zeroed, and a precharge step (automatic call to CHARGE) performed according to CHARGE options if requested.
	<u>CONT</u>	Continue a run, use old densities.
<u>IGICAL</u>	<u>YES</u>	Old style calculation of new GI (expensive). Default for ISTART=NEW.
	<u>SHAD</u>	Shado calculation of new GI.
	<u>NO</u>	No GI calculation (use 1.'s everywhere).
	<u>OLDI</u>	Use old DION's (from previous NTERAK run). Default for ISTART = CONT.
	<u>OLGI</u>	Use old GI's (from previous NTERAK run).
	<u>COPY</u>	Use ion densities from a different set of data file. This option should be used with care. (See section 6.42.30 for details.)
<u>SAVSET</u>	<u>YES</u>	Equivalent of IGICAL = NO overrides IGICAL = YES with exception noted by GISAVE.
	<u>NO</u>	Leave IGICAL = YES alone, SAVSET can be switched at location indicated by GISAVE.
<u>GISAVE</u>	X,L # , I Y,L # , I Z,L # , I	Two fields following the keyword. X 4 means that all nodes in the plane at X = 4, the value of SAVSET will be temporarily toggled. Up to nine planes may be entered.

The combination of IGICAL YES, SAVSET NO, and GISAVE Z -3, GISAVE Z -4, would cause the neutral ion density calculation (GI's) to be calculated everywhere except at Z = -3, and Z = -4. IGICAL YES, SAVSET YES, GISAVE X 4, GISAVE Y 5, GISAVE Z 12, would cause GI values to be calculated at the point 4 5 12 only.

<u>OPTION</u>	<u>VALUE, FORMAT</u>	<u>EXPLANATION</u>
SAVETEMP	ON <u>OFF</u>	Save files 9 and 10 after NTERAK run Delete after NTERAK run.
WHAT		Display NTERAK option settings.
HELP		Activates the NTERAK interactive help.
EFLDCOR	YES <u>NO</u>	Use electric field corrections to the ion density calculations.
IGIOUT	NO <u>YES</u>	Output of appropriate densities.
INPOT	PRE FLOA <u>CONS</u>	Starting surface potentials from a precharge step. Floating surface potentials for first step. Constant surface potentials. Default for ISTART=NEW.
POTVAL	(V_{c1}), E,F	Initial insulator potential. Default is potential of conductor 1 (or V_{c1}).
CONDV	N pot	Initial conductor voltages where N is the conductor number (N=1 is ground conductor) and pot is the initial potential in volts.
CONTCONDV	N pot	Used in continuing run to reset conductor N to <pot> volts (CONDV is valid to first run only).
INSULPOT	DIFF <u>CONS</u>	Defines POTVAL to be the initial DIFFerential voltage from insulator surfaces to underlying conductor or a CONStant insulator surface voltage.
FLOAT	n	Floats conductor n. If n (a conductor number) is omitted, all conductors are floated (the default condition).
FIXP	n V	Fixes potential of conductor n to V volts.
BIAS	n V	Bias potential of conductor n with respect to conductor 1 by V volts.
RIJ	i j r	Sets resistance between conductor i and conductor j to r ohms. For $r < 1$, infinity (also default) will be used.

<u>OPTION</u>	<u>VALUE, FORMAT</u>	<u>EXPLANATION</u>
CIJ	i j c	Sets capacitance between conductor i and conductor j to c farads.
DXMESH	F,E	Grid spacing length in meters.
NXADNT	(0),I	Number of nodes added to the object grid in the +X direction to set up the computational space.
NXADNB	(0), I	Add nodes in -X direction.
NYADNT	(0), I	Add nodes in +Y direction.
NYADNB	(0), I	Add nodes in -Y direction.
NZADON	(0), I	Add nodes in -Z direction.
NZTAIL	(0), I	Add nodes in +Z direction (wake).
IOGRID	NO L YES	Output grid information.
REMARK COMMENT		Comment out. Ignore rest of line.
NPHI	(36),I	Number of zenith angle divisions for GI calculation.
NTHETA	(180),I	Number of azimuthal angle division for GI calculation.
NADD	(2),I	Add extra vertices to object shadow in velocity space for GI calculations.
NPHISH	(16),I	Angular resolution for SHADO module.
NHEXSH	(6000),I	Grid points used by SHADO module.
STHWAKE	ON L <u>OFF</u>	Turn ON or OFF the sheath wake shadowing algorithm.
CHRGENV	(DMSP),L	Select predefined charge environment <name>
VMACH	VX VY VZ,F	Mach vector units of $\sqrt{kT/m}$. Default is set by VEHICL or ORIENT.

<u>OPTION</u>	<u>VALUE, FORMAT</u>	<u>EXPLANATION</u>
DENS	(1.0E10),E,F	Ambient density in α/m^3 .
TEMP	(0.2),E,F	Ambient temperature in eV.
TEMPRAT	(1.0),E,F	T(electron)/TEMP, used only during neutral ion density calculation.
DEN2	(4.2E6),E,F	For energetic Maxwellian in m^{-3} .
TEMP2	(4.3E3),E,F	For energetic Maxwellian in eV.
POWCO	(1.4E12),E,F	Power law coefficient #/m ² .sec.str.eV.
PALPHA	(1.2),E,F	Power law exponent.
PCUTL	(50.),E,F	Power law lower cutoff, in eV.
PCUTH	(1.6E6),E,F	Power law upper cutoff, in eV.
GAUCO	(8.8E5),E,F	Gaussian distribution coefficient #/m ² .sec.str.eV.
ENAUT	(8.2E3),E,F	Gaussian peak in eV.
DELTA	(1.8E3),E,F	e-folding width of Gaussian.
ESECNG	(2.0),E,F	Average electron secondary energy (eV).
RATIH	(1.0E20),E,F	Ion to hydrogen ratio.
AMUION	(16.0),E,F	Ion mass, in AMU
BMAG	(.4),E,F	Magnitude of B field, gauss.
BDIR	(-1 0 0)	Vector direction of B field, normalized by code.
BFIELD	<u>OFF</u> L ON	Turn magnetic field effects on and off.
SUNDIR	(0 1 0)	Vector direction of light source.
SUNINT	(0.0)E,F	Light intensity, use 1.0 for normal solar illumination.
CONVEX	NO YES	Used to turn off surface-surface shadowing calculation when unnecessary.

<u>OPTION</u>	<u>VALUE, FORMAT</u>	<u>EXPLANATION</u>
SHEIONZ	ON <u>OFF</u>	Turns on (off) sheath ionization effects.
NEUTDEN	(1.0E12)E,F	Defines the neutralized ion density ($\#/m^3$).
IONZCROS	(1.0E-20)E,	Defines the interaction cross-section for ionization effects (m^2).
BEAM	ALL ON OFF	Turns on/off all defined beams.
	n ON OFF	Turns on/off beam n.
	beam-char	char-value. Sets beam characteristic of beam n (see 6.42.60).
	ALL	char-value-list. Sets beam characteristics (see 6.42.60).

FOR PWASON

MAXITS	(8),I	Space charge iteration limit.
MAXITC	(20),I	Potential iteration limit.
MINITC	(2),I	Potential iteration lower limit.

<u>OPTION</u>	<u>VALUE, FORMAT</u>	<u>EXPLANATION</u>
RDRMIN	E, F	Potential convergence test. The default value is number of nodes * .0001 volts.
POTCON	(6)	$= \log_{10} (RDOTR(1)/RDOTR(\nu))$, potential convergence test.
IDCONG	<u>PART</u> L NO FULL	Output control of potential calculation.
ISPOUT	PART L FINAL NO FULL	Output control of space charge calculation. To get final potentials from PWASON.
PDIE	(9999)E,F	Maximum allowable positive voltage in screening calculation (volts).
SQALPH	(3),F	Space charge limiting factor.
RMSOFF	<u>ON</u> OFF	Use RMS convergence test to end space charge iteration.
RMSCONV	(TEMP/2)E,F	RMS convergence limit for space charge iteration (volts).
ENORMCHK	<u>NO</u> YES	Check surface boundary conditions after PWASON cycle, redo if any boundary conditions have changed.
MOTIONDEN	<u>ON</u> L <u>OFF</u>	Turn ON or OFF the use of the analytic space charge density.
MIXDEN	(.5),R	Adjust the diagonalization of the finite element charge density. (For pushed particle densities.)
MIXDENMO	(0.),R	Adjust the diagonalization of the finite element charge density. (For analytic space charge densities.)

<u>OPTION</u>	<u>VALUE, FORMAT</u>	<u>EXPLANATION</u>
<u>FOR CURREN</u>		
CURPOT	(-.45*TEMP), E,F	The potential at which the presheath weights are calculated.
STHPOT	PSIM E, F	Particle sheath boundary potential. $PSIM = \ln(SQALPH * (\lambda D / DXMESH)^2)$
WAKSHPOT	-Temp*Vmach**2	Sheath wake boundary potential default = - temp * vmach**2 (volts)
AVEPRTCL	ON <u>OFF</u>	Average the sheath particles found in an element to create one macro particle.
THRMSPRD	ON <u>OFF</u>	Generate a thermal velocity distribution of particles for each sheath particle.
WGTMIN	(1.e-6)E,F	Ignores particles having weight less than WGTMIN.
NTABLE	(10),I	The number of entries in the table used by STHCAL to calculate presheath weights.
IPCNT	(3),I	The maximum number of left and right particle pushing sequences allowed.
IPQUIT	(2),I	Limit of pushing sequences without a change in the number of moving particles.
MAGSTH	ON <u>OFF</u>	Magnetic flux tube reconstruction of electron presheath flux.
RDMAX2	(.25)	Electron drift approximation cutoff.
XYLIMIT	(40),I	Limit to number of elements a particle may enter without leaving Z slice.
STEPLIM	(20),I	Timestep doubled after STEPLIM step pushing steps.
MAXSTPIN	(5),I	Maximum number of timestep doublings due to exceeding STEPLIM.

<u>OPTION</u>	<u>VALUE, FORMAT</u>	<u>EXPLANATION</u>
<u>FOR CHARGE</u>		
SPCLIM		Use space charge limited currents (pushed particles) for attracted species. (Default method)
ORBLIM		Use analytic, orbit limited currents for the attracted species.
ORBSPC		Use analytic, orbit limited currents for the attracted species, normalized by the space charge limited sheath current.
MAXITT	(2) I	Number of timesteps.
DELTAT	(1.0)E,F	Timestep, seconds, default is 1. sec.
DVLIM	(1000.)E, F	Voltage change per step, in volts.
VLTFIX	(V _F),E,F	Estimated potential for non-charging surfaces (Used as a boundary on calculation.) $V_F = -(TEMP/2) \ln(AMUION * m_p / m_e).$
VFIXHI	(-10000),E,F	Estimate of upper (charging cases) crossover potential. Used as a boundary to try and limit voltage swings. Volts.
DVTEST	(5),E,F	Guess used in first CHARGE cycle for voltage change in previous timestep. Volts.
VWIGGL	(.001),E,F	Minimum change forced when a surface is near its equilibrium.
XDVFAC	(2),E,F	Multiplies DVLIM to be used in first trial of a CHARGE cycle to guess the potential of the crossover point.

<u>OPTION</u>	<u>VALUE, FORMAT</u>	<u>EXPLANATION</u>
DVMAX1	(5.*temp)E,F	Value for absolute convergence check (CHARGE) (dV).
DVMAX2	(.05)E,F	Value for relative convergence check (CHARGE) (dV/oldV).
FXFORM	<u>NO</u>	Allow code to choose matrix formulation.
	YES	Force code to use undiagonalized form.
USELIM	<u>NO</u>	See text. An infrequently used keyword.
	YES	

<u>OPTION</u>	<u>VALUE, FORMAT</u>	<u>EXPLANATION</u>
<u>FOR DIAGNOSTICS</u>		
IDIAGS(#) #,I JDIAGS(#) #,I		Diagnostic and output controls, see Section 6.45.10.
KDIAGS(#) #,I IBIAGS(#) #,I		
OUTPUT sec quant		Sets a set of DIAG flags appropriate to subsection (sec) and quantity (quant). Valid subsections are TIMER (run time information), PWASON, CHARGE, CURREN, and NEUDEN (the neutral ion density calculation). Valid quantities are HIGH, LOW, and OFF. Defaults are set to LOW for all subsections.
SELECT options		Used to select specific Z-slices from large tables for output. Also can be used to turn output of a data set off (or on). The various options are described below.
SELECT name list ENDLIST		Name is a buffered name (5.30) which is sliced. list is a set of integers separated by blanks which are object grid Z-slice values (e.g., to get IZ=2 and 3 of POT say SELECT POT 2 3 ENDLIST)
SELECT name ALL NONE OFF		Turns on or off (NONE) output of name (a buffered set of data). Default is on (ALL). OFF resets to default.
SELECT LIST n list ENDLIST		Defines LIST n ($1 \leq n \leq 5$) as list (see above).
SELECT name LIST n		Prints LIST n Z-slices when name data is printed.
SELECT NICE		Prints bit packed lists in a "nice" format, unpacked.
SELECT OCTL SELECT PACK		These both cause packed lists to be printed as they are stored, packed.

6.45.10 NTERAK DIAGNOSTICS AND OUTPUT CONTROL

The diagnostic keywords recognized by NTERAK are IDIAGS(N) and JDIAGS(N). For example, a runstream might contain the card

IDIAGS(9) 1

All diagnostics have a default level of zero. Diagnostic flag levels are usually recognized in a > sense; i.e., larger flag levels include all the lower levels.

A brief explanation of each keyword can be found in Section 6.44.20. The diagnostic flags are as follows:

IDIAGS(1) used by PWASON

- = 1 CONGRD convergence information
- = 2 CONGRD output
- = 3 COPROD output
- = 4 Additional COPROD info, all of the PCUBE routines, and DCVCEL output
- = 5 Additional DCVCEL output

IDIAGS(2) CBUF usage

- = 2 Information from DEBUF, GETLAD, BUFSET (MORCOR calls, IAVECS settings)
- = 3 PAGER (TIMER calls on entry and exit)
- = 4 All info from PAGER and GRIDIO

IDIAGS(3) VERTIO check

- = 4 All VERTIO action

IDIAGS(4) TIMING

- = 1 in OPTIN
- = 2 in CHARGE, IONDEN, CONGRD
- = 3 in PAGER

IDIAGS(5) Self diagnostics

- =2 GRIDIO (calls CHKPUF to check the addressing system).
IONDEN plus others possibly in future (calls CHKLOC to
issue warnings if addresses within CBUF are too big to be
passed within a word.)
- =3 CHKLOC prints core locates of the all locations passed to
it.
- =4 CHKLOC generates non-fatal walkbacks for warning
situations.
- =5 CHKLOC generates non-fatal walkbacks for all calls to it.

IDIAGS(6) Trajectory tracing (CURREN)

- = 1 Trajectory progress checking
- = 2 PRNTSL (particle reading and writing) action, particle
movement at the slice level
- = 3 Complete output from PRNTSL, print out the various grids
used by CURREN on exit
- = 4 Print individual particle movements
- = 5 Energy checking diagnostics

IDIAGS(7) QSELT

- = 4 QSELT information

IDIAGS(8) CHARGE

- = 0 Echo input
- = 1 Control flow info
- = 2 Voltage results from ICCG
- = 3 Input to ICCG, previous cycle data, material info
- = 4 Intermediate subroutine output
- = 5 Surface by surface calculations
- = 6 FLUXEL and FLUXBK output, CHKIMV surface checking

IDIAGS(9) IONCUR

- = 1 Print current totals
- = 2 Intermediate lists, SRFI, SRFH lists
- = 3 Weight calculation results
- = 4 Weight calculations
- = 5 Particle lists from CURRENT

JDIAGS(1) SREL (VEHICL)

- = 0,1 No output
- = 2 Print out LCEL and SREL lists. Also information from RECELL, SREL index to LCEL.
- = 3 Print out preliminary LCEL list
- = 4 Detailed output from routines constructing LCEL and SREL lists.

JDIAGS(2) SHEATH calculation

- = 1 Print tables
- = 3 Print presheath information for particles

JDIAGS(3) PRECHG

- = 2 Print ambient ion currents calculated by AMBCUR
- = 3 ASRF from ALSURF

JDIAGS(4) IONDEN

- = 1 Print progress notes
- = 3 SHFTIT results (for filled or partially filled elements)

JDIAGS(5) Double point data

- = 2 Potentials from POTSET
- = 3 On input to major modules and double point locations from SPDPNT (vehicl)
- = 4 More double point information from SPDPNT (vehicl)

JDIAGS(6) RHOI values

- = 2 Print RHOIs after completion of CURREN (in CUEXIT)

JDIAGS(7) Particle pushing error control
= 1 Print info for particle in error
= 3
= 4 Kill run if 5 percent of particles are lost to errors
= 5 Die instantly when an error occurs

JDIAGS(8) Modified CHARGE output
= 1 Final surface voltage for the cycle, including INISET
= 2 Initial FTOT and FSM used (total current and total secondary current)
= 3 Components of current for each surface

JDIAGS(9) MTLGEN, GENMTL (in VEHICL)
= 2 Print calculated AMAT in MTLGEN
= 3 Print GENMTL's LIST, CMAT
= 4 Print all appropriate information

JDIAGS(9) IONGEN (in NTERAK)
= 2 IONGEN prints DIONs
= 3 IONGEN prints LTYPs

KDIAGS(1) MSMODS
= 1 Routines echo calls to themselves

KDIAGS(2) CHKSTH, CHKTRJ
= 3 Echo commands

KDIAGS(3) E field correction to geometric ion densities
= 2 Print tables which will be used
= 3 Print E field correction self check

KDIAGS(4) DBLCHK (particle validation)
= 2 Potentials for rejected particles
= 4 Position velocity for all particles

KDIAGS(5) Action on error
 = 0 Print simple error message
 = 1 Print arguments, variables, etc.
 = 2 Print arrays, lists, slices
 = 3 Go to interactive dump (FTNPMD on UNIVAC)

KDIAGS(6) Photosheath
 = 0 None
 = 1 Rotation angle
 = 2 Shadowing surface inversion list creation
 = 3 Intermediate information
 = 4 Produce hidden line plot

KDIAGS(7) E field boundary on surface
 = 0 None
 = 1 Start or end of PWASON
 = 2 SURFV's and IFLT at start (and end)
 = 3 Start and end of each space charge iteration
 = 4 Also SURFV's at each step
 = 5 E parallel for each step

KDIAGS(8) Edge list calculation
 = 0 None
 = 1 Welcome and timing
 = 2 Results
 = 3 Intermediate information.

KDIAGS(9) FIX/FLT surface determination
 = 0 None
 = 1 PWASON fix/float interface
 = 2 CHARGE fix/float calculation overview
 = 3 CHARGE fix/float calculation defaults

LDIAGS(1) Shado sheath information
= 1 Brief hex grid information
= 2 More information of hex grid construction.
Create fort.12(PLOTXY) and fort.50(NPLDIS)
= 3 Detailed hex grid construction (individual nodes)
= 4 Detailed neutral ion densities calculation at each node

LDIAGS(2) Particle trajectory stuff
= 0,1 No output
= 2,3 Particle trajectory information from PUSHER

IBIAGS(1) Magnetic field diagnostics from CURREN
= 0 None
= 1 Some
= 2 More
> 2 Lots

IBIAGS(2) Magnetic field information from CHARGE
= 0 None
= 1 Some
= 2 More
> 2 Lots

6.50 OPERATING SHONTL

SHONTL is POLAR's general plotting and information extraction program. Its primary sources for these functions are the two mass storage files, 11 and 19. For certain functions, SHONTL will actually duplicate NTERAK calculations. It should be noted that SHONTL remembers the settings from previous uses on the current data files. This means that some care should be taken to check the current flag setting.

SHONTL's primary plotting mode is to construct two-dimensional contour plots of data slices perpendicular to any of the three coordinate axes. Currently, the following variables may be viewed in this fashion: neutral ion densities ('GI's and 'GH's), composite ion densities ('DION's and 'DELC's), stabilized charge densities ('QUSD'), and potentials ('POT'). In addition, SHONTL can execute the CURREN module of NTERAK and superimpose the results on any of the contour plots. In particular, there are two types of graphical output from SHONTL-CURREN. The sheath location can be indicated by a contour of X's using either the sheath potential that was used by NTERAK-CURREN, or optionally changed with the STHPOT keyword (Section 6.52); also, ion trajectories may be displayed with the following options: A total perspective of complete trajectories projected onto the plot's slice (keyword PERSPC), the portion of all trajectories that pass through the plot slice, trajectories emanating from a ring of sheath points lying in selected X-Y plane (X-Y only) (keyword RING) which can also use the PERSPC option, and in addition to all of the above the WEED option will throw out about 75 percent of the trajectories to reduce plot cluttering and computing costs. Finally, a single point trajectory option is also available (keyword SPOTS).

Other plotting features are limited to a spectrum plot of the energetic electron environment. Future modifications will allow SHONTL to reproduce the material plots and the hidden line object perspective views that are currently available only from VEHICL.

In addition to plotting, SHONTL can output to a printer (or whatever) the current values of virtually any variable stored on files 11 and 19, through the use of the PRINT NAME keyword where NAME would be the exact FORTRAN name of any variable known to MRBUF (a subroutine). A list of these can be found in Section 5.33. The format for this output is generally identical to that used by VEHICL, ORIENT, and NTERAK when these variables are output from these modules. A note of warning - variable aliasing is not always identical between modules. For example, potentials would be output by NTERAK by the 'card'

```
ISPOUT FINAL
```

which would produce potential array output (modified by the SELECT keyword) at the conclusion of a spacecharge-potential calculation (the PWASON submodule of NTERAK). Whereas in SHONTL one would enter the cards

```
SELECT - options -  
PRINT POT
```

to get the same output.

Like the other modules, SHONTL keywords fall roughly into two categories that can be described as verbs and adverbs. For example, a user might enter the following adverb cards to set up plot parameters:

```
PLOTS ON  
LEVELS 20  
LEVELS NOMARK  
GRDPTS ON  
SHEATH ON
```

followed by the verb card

```
POT X
```

which would trigger the actual production of a potential contour plot of the slice through the center of the object grid perpendicular to the x axis with a sheath indicated, no contour markings, approximately 20 contour levels, and all grid points drawn.

Finally, plots (but not output) are generated in a machine independent fashion on file 2., and a post-processor (usually PLOTREAD) is used to interface to the local graphics package. Depending upon installation and requirements, the user might want file 2 to be a permanent file.

6.51 STEP BY STEP INSTRUCTIONS FOR SHONTL

Step 1: Two data files are needed by SHONTL. They are file 11, and these files must be named 11 and 19. The best way to do this on the UNIVAC is to use the QUSE command. Note: If RHOI's and RHOE's are to be examined, file 9 is also needed (SAVETEMP must be ON during NTERAK run).

Step 2: Execute SHONTL in an interactive mode.

Step 3: Now you need to enter the information needed to retrieve the slice you wish to plot. The instructions for the input routine are in the form of keywords (6.52). Additionally, there are several commands to help use the keywords to get the desired plots (6.52).

The first step is to choose the plotting features you wish to use. These include marking the grid boundaries, showing the silhouette of the object, marking the grid points, choosing the axis for the problem and choosing a title for the plot. (There is a complete list of the keywords in the next section.)

Step 4: When all of the desired features have been chosen, a slice needs to be defined. At this point a plot is generated. There are several useful defaults which produce sets of plots (see 6.53). Now one can return to step 4 to produce additional graphs or continue onto step 5.

Step 5: To leave SHONTL, type EXIT if you want hard copies of the plots to be generated. Type ESC (for escape) if no hard copies are desired. The EXIT command will call PLOTREAD automatically.

Step 6: Show all of your friends your nifty plots.

6.52 SHONTL KEYWORDS

The following is the current list of SHONTL keywords. All of the input which cannot be recognized is simply printed on the screen and then ignored.

The form of a keyword definition is

KEYWORD[, equivalent forms](*default setting*/other settings)
definition

The terms "ON" and "OFF" describe the setting of the option. So if something is set to "OFF", it will not appear on the plot. For example,

PLOTS OFF

will cause those plots generated while the flag is off, to not be printed. Since default value of PLOTS is off, to prevent undesired plots, the definition of PLOTS looks like

PLOTS (*OFF*,ON) causes the

Here is the present keyword list:

ALL (ON/OFF) turns all of the on/off switches to ON or OFF. "ALL ON" would turn on all of the keywords which can be set to (ON/OFF).

AXDRAW (*ON*/OFF) controls the printing of the plot axis, scale values, and axis labels.

A2SURF (*ON*/OFF) controls the printing of the object's silhouette.

BEAMTRAJ (ON/*OFF*) causes particle beam trajectories to be made instead of the sheath particle trajectories which are usually printed when TRAJ is ON.

CHEAT. This keyword sends the user to an input subsection which can be used to change data in the mass storage files. The changes made here are permanent, so this should be used with care. A new value can be entered into the list type buffered data variables (for example SURFV), but none of the spatial, or sliced, data sets. The general form is

name entry value

where name is the list name (see 5.33 for a complete list), entry is the entry to be changed, and value is the number to be placed in the entry location. The entry is either an integer or the keyword ALL.

SRFI ALL 0.01

This command would set all the entries for SRFI to .01. Only one command will be accepted, then the user is returned to the normal input mode. This input is intended for use as a developmental tool and is not user friendly. There are no array boundary checks. If escape is desired after the CHEAT keyword, enter

ABORT

and the normal input mode will resume. Be careful when using CHEAT.

COLRLINS. Used to draw colored line plots. An input subsection is entered where contour lines (CONL), A2 surfaces (A2SU), grid boundaries, points, and axes (PLOT), the sheath edge (SHEA), and particle trajectories (TRAJ) can be black (or white depending on the background color, BLAC), green (GREE), red (RED), or blue (BLUE). For example, if green particle trajectories the input would be

TRAJ GREEN

The keywords will be truncated to four letters. To leave the subsection and return to the normal input mode, enter EXIT.

COLOR (ON/*OFF*) turns on color filled features. This may depend on the use of a Jupiter.

COLORDIV (ON/*OFF*). When ON and COLOR is also ON, this keyword draws white contour lines between the color levels on the plots.

COMMENT causes the input routine to ignore the rest of the line. Note that only the first six characters on the line are actually read.

CONLIN (*ON*/OFF) controls the printing of the contour lines on the plot.

DEBYE causes the scale on the plot to be in Debye lengths. The origin (0,0) is the object grid origin (see 'GRID'). The normal default is to grid units.

DION,DENSIT,DEN,D. This is a request for a plot of a slice of ion density data (DION's). The allowable forms of requests for density slices are

```
DENSIT X N
DENSIT X MIDDLE
DENSIT X
DENSIT
```

where either of the abbreviations may be substituted for DENSIT. Also Y or Z can be used for X, MID can be used for middle, and N is a positive integer on the axis (X, Y, or Z) that is in the grid space. The axis is the axis perpendicular to the plane of the slice. If form 2 or 3 are used, the middle of the object grid will be plotted. If the last form is used, the middle X and Y slices will be plotted.

DELC is similar to DION but electron density data is plotted instead.

EDGESTOP (NO/*YES*). Normally particles are pushed until they hit the object, exceed an iteration limit, or go outside the sheath boundary. Sometimes, like when pushing beam particles, this is undesirable and particles should be allowed to pass the sheath edge (NO option). The outer boundary on pushing then becomes the grid boundary.

ERASE causes any plots which have been created to be erased and not produced by PLOTDR.

ESC causes the SHONTL module to be exited immediately without any additional output and without resetting the previous defaults. (The keyword ESCAPE is equivalent.)

EXIT signals the end of the input. If no plots have been requested at this point and 'PLOTS' is 'ON', then four plots will automatically be produced. They will be plots of the potential and density at the middle of the X and Y axes.

FILLIN (*OFF*/ON) causes the interior of the object to be filled. The routine looks for zeros surrounded by relatively big numbers. This changes the appearance of the contour lines in a way which may be desirable. (Obsolete.)

FIXOPS. When the default SHONTL values may not be set right, this allows the user to force the default values to be set again.

FRAME (*PLASma*/CRAft) - (**/X Y Z). This keyword chooses the reference frame for viewing potentials. The default is the plasma frame used by NTERAK, but a spacecraft frame may be chosen where the $\vec{v} \times \vec{B}$ electric field will appear at 'infinity'. In the spacecraft frame the definition of plasma ground will appear arbitrary (it is not really) but it may be shifted from the point chosen by NTERAK.

GETSLC (*ON*/OFF) causes the plot generation routine to read a slice from the data file.

GI and GH as DION, above, except geometric ion densities and hydrogen densities are plotted instead.

GRDPTS (ON/*OFF*) causes crosses to be printed at each of the nodes.

GRID causes the scale of the plot axes to be in object grid units. The reference point in this coordinate system is the lower left corner of the object definition grid which is defined to be (1,1). This is the default scaling.

HEADNG (*ON*/OFF) causes the heading to be printed at the top of the plot.

HIDPLOT (*ON*/OFF) causes a hidden line drawing of the object to be created. See also VIEWDIR in this section.

IDIAGS(N) is the keyword input section which allows setting of any of the IDIAGS flags. For information concerning the IDIAGS, see Section 6.70.

INPSPT (*0*). The keyword, INPSPT (INPUT SPOT), is used to trace the trajectory of a number of particles (SPOTS) which the format for keyword is "INPSPT N", where N is the number of points to be entered. The maximum number of points is 20. After the INPSPT keyword, the N particles are defined by giving their initial position (X,Y,Z), velocity (VX,VY,VZ) and the particle (ELEC for electric type and ION for ions). For example, if we wanted to define two particles, normalized to the electron acoustic speed, with the position (4.,5.,6.) in grid units and velocity (-.1,-.1,-4.) normalized to the ion acoustic speed (see documentation, POLAR Manual, Section 5.62.12), and a second particle, on electrons, with the position (-1.,3.,-2.5) and a velocity of (-1.,-.3,.01), normalized to the electron acoustic speed, we would enter:

INPSPT 2

4. 5. 6. -.1 -.2 -4 ION.

-.1 -.3 -2.5 -1. -.3 .01 ELEC

The default particle type is ION.

IOGRID prints the nuts and bolts mesh grid information printed by NTERAK using the same keyword (see 6.44.20).

JDIAGS(N). See entry for IDIAGS(N).

KDIAGS(N). See entry for IDIAGS(N).

LEVELS (*10*-AUTO) set the number of contour lines to be drawn and their selection mode. The keyword LEVELS is also an entry word to the following contour options:

LEVELS AUTO linearly spaced contours, rounded to nice numbers, for potentials, the kT and $0.1*kT$ contour will be added and the kT level marked.

LEVELS NOMARK turns off all contour marking.

LEVELS MARK VALUE MARKER will mark the contour level at VALUE with the first character of MARKER. Marking with a number can be accomplished by #LLL where # is a number and L is any letter. Only # will be used, but the LLL is necessary for # to be recognized as a literal. This command may be repeated to mark up to nine levels.

LEVELS ADD VALUE, add a contour level

LEVELS SUB VALUE, remove a contour level

LEVELS SELECT VALUE, turn off auto mode and use input levels

LINEAR means the contour lines will be of the unmanipulated data in the slice. This is the default mode.

LOG causes the contours to be of the natural log of the slice data. The normal default is 'LINEAR'.

LOG10 causes the contours to be of the base 10 log of the slice data. The normal default is 'LINEAR'.

METERS causes the units of the scale of the plot axis to be in meters. The origin (0,0) is the object grid origin (see 'GRID'). The units default to grid units.

MIDDLE,MID. These keywords must follow X, Y, or Z to be meaningful. In this context, they cause the plot to be of the middle of the object grid. (For a more exact description of the middle of the grid, see 'DENSITY'.)

MKCHEAT. This is used to generate input for the CHEAT command.

MSINDEX N. This keyword is used to look at the index of the random access mass storage files created using the MSIO package. N is the file of interest, there is no default.

NONE. This keyword can be substituted for POTENT or DENSIT. It will cause plots to be made without potentials or ion densities. It can be used to look at the output from VEHICL. When plots of sheaths and trajectories are made, it is also helpful. The syntax for NONE is exactly the same as for POTENT or DENSIT.

NTEWHAT. Print the current values of NTERAK variables. Equivalent to saying WHAT while running NTERAK.

NUMBER (ON/*OFF*). NUMBER is used when tracing particles. If it is used in conjunction with IDIAG(6)=5, the particle pusher will number the particles for diagnostic output. This is very useful for debugging the CURREN segment. (See also the notes on IDIAGS.)

OBJCNR (ON/*OFF*) causes four small boxes to be drawn in the corners of the object grid.

OBJGRD (ON/*OFF*) causes the object grid to be plotted.

OUTREL (*ON*/OFF) causes the outer stepped grid of real nodes to be plotted.

OUTVIR (*ON*/OFF) causes the outer stepped grid of virtual nodes to be plotted.

PERSPC (ON/*OFF*). When PERSPC is on, all of the sheath points and trajectory paths will be plotted. If it is off, then only those points or paths which are in or cross the slice being drawn will be shown.

PLOTS (ON/*OFF*) causes the plots to be produced. If it is off, the slices will be read and processed for plotting, but will not be plotted.

PLTGRD causes the scale of the axis to be in plotting grid units. These are the same size as object grid units but the lower left corner is defined to (0,0). In object grid units, the reference point on the grid is the lower left corner of the object definition grid which is defined to be (1,1). The default is object grid units.

POTENT,POT,P. This is the same as 'DENSIT' except potential slices instead of density slices will be plotted. Please see 'DENSIT' for more information.

PRINT NAME causes NAME to be read from the appropriate file. Valid NAME replacements are any of the buffered data lists mentioned in Section 5.33. For larger data sets, the use of SELECT is recommended (see below).

QUSD. This is the same as the DION (above) keyword except a plot is made of the QUSD data (space charge used by PWASON).

RESET. This keyword resets all of the plotting flags to their initial values.

RESTART is similar to RESET except the previous set of default values left in SHONTL are reinstated instead of the standard defaults.

RING (ON/*OFF*). Sometimes on large problems, it is desirable to trace fewer particles. One way to do this is just follow a ring of particles. Presently the only rings in the XY plane can be made. N is the Z slice of the ring. If a ring in the Z=22 slice was desired, the keyword would be "RING 22". If RING is followed by ON or nothing at all, the middle Z slice will be used.

SELECT options used to choose particular Z-slices for printing when using PRINT or a DIAG (or IDIAGS, etc.) flag. Many options are available and the user is referred to Section 6.44, 6.45 or 6.70.

SHEATH (ON/*OFF*) causes the sheath points to be printed on the plot.

SPOTS (ON/*OFF*). To push a few test particles to see which trajectories they will take, use the SPOTS flag. To enter these particles, see INPSPT. It is recommended that PERSPC is ON when SPOTS is used.

STHPOT N. The SHEATH potential for plotting the sheath can be changed by resetting STHPOT. The initial value is the same as the value used by CURREN during NTERAK. N is a real number.

TITLE. Used to define the title to be put on the plot. The default is "POLAR PLOTS". The desired plot title should start in the ninth column of the card

```
1234567890123456
```

```
TITLE  BIG BIRD
```

In the above example, the heading appearing on the plot would be "BIG BIRD". Currently there is a 16 character maximum.

TRAJ (ON/*OFF*). This flag turns on plotting of particle trajectories.

VIEWDIR. This keyword is used to determine the view direction for a hidden surface perspective plot. For example,

```
VIEWDIR  2.0  2.0  0.0
```

would define a view direction between the positive X and Y axes on the XY plane. The viewing vector does not need to be normalized. This keyword behaves the same as the NTERAK keyword SUNDIR (6.42.10), except the default value is the last value of SUNDIR. The pictures produced by this command will be in color or black and white depending only upon the device driver (TEKLOT or JUPITER) used to display them.

WEED (*ON*/OFF). This flag reduces the number of particles usually found by CURREN by about a factor of 4. This not only speeds up the plotting process but also makes the plots a little clearer.

WHAT causes all of the print options and their current settings to be printed at the terminal.

X,Y,Z the axis perpendicular to the slice to be plotted. If both potential and density plots are desirable, something like

X 8

can be typed. This plots the X = 8 slice of the density and the potential data. Instead of 8, any integer containing part of grid, middle or mid, or nothing at all can be used. Nothing at all will default to a middle slice. Valid forms are

X N

Y MIDDLE

Z MID

X

where X, Y, and Z are interchangeable and N is any integer as described above. Please see 'DENSIT' for more information.

6.53 SPECTRUM KEYWORDS AND OPERATING INSTRUCTIONS

The spectrum plots are generated by SHONTL using the plasma characteristics used by a previous NTERAK run or by defining the environment using the same keywords used in NTERAK.

SHONTL keywords:

(Note that the range of values plotted on the vertical axis is currently not controllable by the user and is limited to six orders of magnitude down from the maximum flux while in the LGLINY LOG mode.)

Default options are marked by surrounding *'s. These *'s are not part of the keyword.

KEYWORDS

HIGHEN	n	n is the value of the energy cutoff on the high end.
INKEV	NO *YES*	y axis always in eV, x axis is variable.
LGLINY	log *lin*	a log 10 or linear plot of data on the vertical axis.
LGLINX	log *lin*	a log 10 or linear plot of data on the horizontal axis.
NPTS	n	number of points to be calculated and connected ($0 < n < 200$).
RLOWEN	n	n is the value of the low energy cutoff on the horizontal axis.
SPECTRUM		draw a spectrum plot using the previously entered information.
WRITE		write the plasma environment generated so far during the current run onto files 11 and 19.

6.54 SHONTL DEFAULTS

SHONTL has lots of defaults. All of the plot features have defaults (6.52). There are also slice default values for which slices are printed. The current slice defaults are also in Section 6.52 under the 'DENSIT' definition.

The default values will vary as the plotting requirements change. The best place to look for default changes will probably be the code itself. The WHAT command gives the feature defaults.

6.55 SPECIAL SHONTL OPTIONS

In addition to two dimensional line drawings produced on file 2, some other forms of graphical output are available. To display these files, it is necessary to use graphics package not usually supplied with the POLAR package. The supported file formats are compatible with MOVIE.BYU (Brigham Young University), NPLDIS (S-CUBED in-house code), and IRMA (AFGL in-house code).

The files are in an ASCII format and straight forward to read. If one is interested in translating the three dimensional graphics on these files to some other package, here is the format of the NPLDIS files. There are two types of files, trajectory files and polygon oriented files. These files are free format files and should be read with "read *" (in fortran) or some free format input package.

Trajectory files contain line segments for each step for each particle pushed. The format is:

```
x0 y0 z0 area
x1 y1 z1 area
```

where x0, y0, and z0 are the 3 space coordinates of the particle before it is moved and x1, y1, and z1 define the final position of the particle. The area is either the flux associated with the particle or a number used to identify it (if the SHONTL NUMBER keyword was used).

Polygon oriented files simply contain a list of polygons. Each polygon follows the following format:

```
Num_Corners Value_1 Value_2 .... Value_Num_Corners
X_Corner_1 Y_Corner_1 Z_Corner_1
X_Corner_2 Y_Corner_2 Z_Corner_2
```

```
.
.
```

```
X_Corner_Num_Corners Y_Corner_Num_Corners Z_Corner_Num_Corners
```

where the Num_Corners is the number of corners on the convex polygon, Value_1 to Value_Num_Corners are the values at the corners, and the following cards define the three dimensional locations of the corners. The corners are defined counterclockwise with respect to the outward surface normal. If only the first value is defined, it is assumed to be the value of all of the corners.

The following is a list of SHONTL keywords and their defaults which only apply to these enhanced graphic capabilities.

BYU (ON/*OFF*) Print the output in MOVIE.BYU file format.

FACEVALU (ON/*OFF*) Set the surface corner values to the surface center value.

NPLDIS (ON/*OFF*) Print the output in NPLDIS input format.

SURFPLOT (vprop name) Make a surface plot of the named vector property. The property must be a surface list item, like material type (IMAT), surface potential (SRFV), or one of the surface currents.

3DCONT Enter the three dimensional spatial potential contour value mode. The following keywords are valid while in 3D Contour mode.

QUIT	- make output file and return to SHONTL
HELP	- this information
WHAT	- list current settings
FILENAME	- prompts for new output file name
ADD <real>	- add contour level
REMOVE <integer>	- remove contour number #
ESCAPE	- leave without plotting

6.60 PLOTTING UTILITIES

The final step necessary to produce the picture created by any of the POLAR modules is the use of a plot driver. Presently there are two plot drivers available for the UNIX version. They are T4014 and PSTPLT.

The first, T4014, is the more commonly used since it converts the general graphics data produced and saved on the file fort.2 by the POLAR modules to Tektronix 4014 graphics code. Since most graphics devices can emulate a Tektronix 4014, this will draw pictures on most graphic terminals.

PSTPLT translates the contents of file 2 to Postscript.

READ02 converts the contents of file 2 into ASCII. The file 4, created by READ02, can be turned into Tektronix 4014 code by running PLOT04.

To draw pictures, move to the directory containing the fort.2 file you wish to plot. Type

T4014

to execute the tekplot absolute. After each plot, the program will wait for the return key to be hit before drawing the next picture.

On VMS systems, it is necessary to issue the following command before attempting to plot.

set term /nowrap

6.70 Surface Charging Utility

A surface charging utility, SUCHGR, is included in the POLAR package. SUCHGR calculates plasma currents and equilibrium potentials for surface materials in orbit limited and space charge limited charging environments. No magnetic field effects, geometric effects or surface to surface interactions are modeled. The tool can be used before beginning an NTERAK calculation to determine the dominate charging mechanism, an estimated equilibrium potential, and the space charge sheath radius.

SUCHGR searches for a surface voltage where the total current to the surface is zero. There may be more than one such point, so the starting potential may be important. Two different algorithms may be used to calculate the portion of current due to the attracted species, an orbit limited collection formula or a space charge limited formula. The physics of these collection mechanisms is shown below in figures 6.70/1 and 6.70/2.

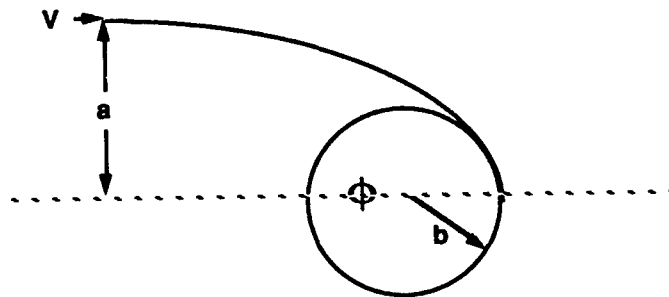


Figure 6.70/1. Orbit limited collection of a particle of mass m , charge q , and initial velocity V is collected by a sphere of radius b at potential ϕ . The maximum impact radius, a , that the particle will be collected is determined by conservation of angular momentum and energy and is

$$a = b \sqrt{1 + \frac{|q\phi|}{\epsilon}} \quad \text{where } \epsilon = \frac{1}{2}mV^2.$$

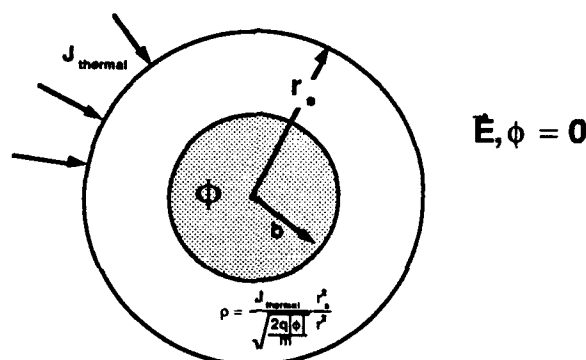


Figure 6.70/2. For short Debye lengths space charge shielding dominates over angular momentum the current collection physics. As shown in the figure a sheath forms outside of which the plasma is undisturbed and inside of which the plasma density is related to the geometry and potentials. For this case Poisson's equation must be solved self consistently with the determination of the charge density. For a planar geometry an analytic solution exists giving the distance from a plate at potential ϕ

to the edge of the sheath, $S = \sqrt{\frac{64\pi}{81}} \lambda \left(\frac{\phi}{T} \right)$. For

spherical geometry a closed form solution is not known; however, the solution can be written in terms of a universal function which can be determined numerically. This universal function, f , can be used to find the

sheath radius through $\frac{r_s}{b} = f\left(\frac{S}{b}\right)$.

To use SUCHGR, define the material type and the environment. Surface material types may be defined by using the predefined material names discussed in section 6.12 or by defining or modifying each material property individually. A complete listing of SUCHGR keywords and their defaults is included below. Plasma environments are defined using the same keywords used to define NTERAK environments. These keywords are discussed in detail in section 6.42.10. Some complete default environments have also been predefined.

After defining the surface material and environment by hand or by running VEHICL, define the initial potentials, the magnitude of the Mach velocity, and the size of the object radius. Finally, select the desired algorithm for calculating the flux of the attracted species and enter the CHARGE keyword.

At any time, a list of the available keywords and a short description of their functions can be printed by entering the HELP keyword. To leave SUCHGR, enter EXIT. The SHOW keyword can be used to view the keywords used to define surface materials, environments and the charging parameters, as well as a short description of the keywords and their current values.

The following is the result of entering the following keywords as input to SUCHGR. Input keywords:

HELP

SHOW ALL

QUIT

And the output:

Welcome to SUCHGR 1.3

Default material is KAPT

Default environment is DMSP

SUCHGR command >> HELP

Available Keywords are:

EXIT		Exit SUCHGR
DEFA		Reset to default setup
SHOW	opt	Show current settings of 'opt' (SETUP, ENVI, MATE, or ALL)
TABLE	opt	Print 'opt', in tabular form (IV, YIELD, ESEC, EBAK, ISEC, MATE)
PLOT	opt	Plot 'opt' (same as in TABLE)
SAVE	fn	Save environment to file <fn>
READ	fn	Read environment (saved by SAVE keyword) from file <fn>
FILE	fn	Read Keyword input from file <fn>
CHARGE		Start charging
CURBRK	ON/OFF	Print Flux breakdown after charge

Use SHOW 'opt' for more keyword descriptions (e.g. 'SHOW MATE' for Material keywords)

SUCHGR command >> SHOW ALL

Charge Setup Keywords & Current Settings

Keyword	Description	Values	Units
SRFVLT	Initial Material Potential	0.0000e+00	volts
CNDVLT	Initial Conductor Potential	0.0000e+00	volts
SUNINT	Solar Intensity	0.0000e+00	(between 0 & 1)
VERROR	Fractional Error in Potential	3.0000e-02	(none)
EPSCUR	"Zero" Current	1.0000e-10	amps
FLOAT	Float Conductor Voltage	(none)	
VBEGIN	Begin Potential for IV Table	-5.0000e+01	volts
VEND	End Potential for IV Table		
AVMACH	Magnitude VMACH	8.0000e+00	(none)
ROBJ	Radius of Object	1.0000e+00	meters
ORBLIM	Orbit Limited Regime	ON	(none)
SPCLIM	Space Charge Limited Regime	OFF	(none)
MATE	Selected Material	KAPT	(none)
ENVI	Selected Environment	DMSP	(none)

Environment Keywords & Current Settings

Keyword	Description	Values	Units
ENVNAM	Environment Name	DMSP	(none)
AMUION	Ion Mass	1.6000e+01	AMU
RATIH	Ion to H+ Density Ratio	1.0000e+20	(none)
TEMP1	Ambient temperature	2.0000e-01	eV
DEN1	Ambient density	1.0000e+11	#/m**3
POWCO	Power law coefficient	1.4000e+12	#/m2.sec.str.eV.
PALPHA	Power law exponent	1.2000e+00	(none)
PCUTL	Power law lower cutoff	5.0000e+01	eV
PCUTH	Power law upper cutoff	1.0000e+06	eV

TEMP2	For energetic Maxwellian	4.3000e+03 eV
DEN2	For energetic Maxwellian	4.2000e+06 #/m**3
GAUCO	Gaussian distribution coef.	8.8000e+05 #/m2.sec.str.eV.
ENAUT	Gaussian peak	8.2000e+03 eV
DELTA	e-folding width of Gaussian	1.8000e+03 eV

Material Properties Keywords & Current Settings

Keyword	Description	Values	Units
MATNAM	Material Name	KAPT	(none)
DIELEC	Dielectric Constant	3.5000e+00	(none)
THICK	Thickness	1.2700e-04	meters
CONDUCT	Conductivity	1.0000e-16	MHO/M
ATOMNUMB	Atomic Number	5.0000e+00	(none)
DELTAMAX	Delta Max	2.1000e+00	(none)
EMAX	E-Max	1.5000e-01	keV
RANGE1	Range_1	7.1480e+01	angstroms
EXP1	Exponent_1	6.0000e-01	(none)
RANGE2	Range_2	3.1210e+02	angstroms
EXP2	Exponent_2	1.7700e+00	(none)
PROYIELD	Yield for 1keV Protons	4.5500e-01	(none)
PROMAX	Max de/dx for Protons	1.4000e+02	keV
PHOTOCUR	Photo Current	2.0000e-05	amps/meter**2
RESIST	Surface Resistivity	1.0000e+16	ohms
SPDISCHR	Space Discharge Potential	1.0000e+04	volts
INDISCHR	Interal Discharge Potential	2.0000e+03	volts
RICCOEFF	Radn-Induced Cond. Coeff	1.0000e-13	MHOMS3
RICPOWER	Radn-Induced Cond. Power	1.0000e+00	(none)
MATDENS	Density	1.0000e+03	kg/m*3

SUCHGR command >> QUIT

6.80 TRMTLK

TRMTLK is an interactive program for retrieving POLAR data. It accesses POLAR restart files to produce charging history tables and graphs and a variety of other information. Charging history information is stored in file 16.

TRMTLK is a menu-style program--the user requests the desired information rather than answering a series of yes or no questions, "Do you want such and such? How about such and such? ..." As a result, the user can get any of the types of available output just by typing one or two commands, no matter what section of the program is currently operating.

There are four main program modules, named HISTORY, LATEST, SINGLE, and SPECIAL.

The HISTORY module outputs for any surface cell the time history of the user's choice of five qualities: potential, electric flux, external electric field, internal electric field (stress), or potential difference between an insulator surface and underlying conductor (delta). The output comes in the form of a printed table, and/or a rough plot of quantity versus time. The printed table will show up to seven surface cells across the page.

The LATEST module gives the user a complete list for all of the surface cells and conductors of any of the five quantities mentioned above. They may be printed in sequence or ordered by magnitude. Partial listings, including only some surface cells, are available.

Module SINGLE prints out information about a single surface cell. In addition to the five standard quantities, you can get the cell location, its surface materials, its surface area, its shape, its normal, and the number of its underlying conductor.

The SPECIAL module lets the user "turn off" all output coming to the terminal. This unseen output can be printed on a line printer at the end of the session. SPECIAL also allows the user to change the POLAR cycle number.

6.81 PROGRAM STRUCTURE

TRMTLK is organized into four modules: LATEST, HISTORY, SINGLE, and SPECIAL. Each of the modules has its own set of commands. Each module also has access to a set of "aid" routines, named HELP, SUBSET, OUTLINE, LOCATION and EXIT.

The user moves from any module to any other by typing the name of the new module. "Aid" routines always return to the module from which they were called.

6.82 CELL SPECIFICATIONS

POLAR assigns a cell number to each surface cell. Each cell can also be identified with five basic geometrical attributes: location, material, surface normal, shape, and conductor number. TRMTLK provides a way for the user to go back and forth between cell numbers and geometrical attributes.

"SINGLE" takes a cell number and prints out attributes. (it also prints out charging information.

"SUBSET" takes the user's geometrical specifications, and determines the set of cells that satisfy these. It is extremely general. You can easily determine the number of a particular cell, or you can define a complicated group of cells. For example, you could define the group of all nonboom cells that are either KAPTON or TEFLON and lie on conductor #1; or all cells between the planes of $X = 1$ and $X = 5$.

6.83 CHARGING HISTORY

TRMTLK can plot surface cell potential as a function of time. It also prints cell histories in a tabular form. And it gives tables and plots of other quantities such as electric flux and electric fields.

If you are more interested in final equilibrium values than in histories, the module "LATEST" gives only the most recent information.

6.84 INSTRUCTIONS FOR USE

The most extensive documentation for TRMTLK is internal to the program. It is accessed by typing "HELP" at any time. This gives the user a full-blown explanation of the commands and modes available for each module.

It is necessary to run POLAR before using TRMTLK. A simple object definition will initialize enough files to use SUBSET. For instructions on POLAR use, see the POLAR User's Manual.

To use TRMTLK on a UNIVAC computer, type 'OXQT TRMTLK.ABS'. The program will ask for the POLAR file prefix and assign all necessary files.

On CDC machines, the user must attach POLAR files 11, 16, and 19 before running. At the end of each run, file number 3 will contain a line printer image of the output.

6.85 SAMPLE RUN

The following shows a simple TRMTLK example. Note that near the end, the mode 'NOTERM' is set. This suppresses terminal printing of output data. The user will see on line printer output, without having to wait for terminal printout of the HISTORY graph.

Welcome to POLAR TRMTLK ...

Any AID may be called from any MODULE

MODULES

AIDS

HISTORY

AGAIN

LATEST

HELP

SINGLE

LOCATION #

SPECIAL

OUTLINE

SUBSET

EXIT

SUBSET [GROUP NAME]

Enter any MODULE/AID name or 'HELP' for help >> single

SINGLE command or MODE set >> 15

SURFACE NO. 15
MATERIAL IS GOLD

CENTERED AT 4.50 6.50 5.50

POTENTIAL = -2.3562e+01 VOLTS

SINGLE command or MODE set >> also stress
MODE RESET

SINGLE command or MODE set >> also delta
MODE RESET

SINGLE command or MODE set >> 5

SURFACE NO. 5
MATERIAL IS KAPT

CENTERED AT 5.50 5.50 5.00

POTENTIAL = -5.3642e+00 VOLTS

DELTA V = 1.8198e+01 VOLTS

INTERNAL FIELD STRESS = 1.4329e+05 VOLTS/METER

SINGLE command or MODE set >> every MODE RESET

SINGLE command or MODE set >> 20

SURFACE NO. 20
MATERIAL IS KAPT
SHAPE IS RIGHT TRIANGLE

CENTERED AT 6.33 4.67 6.00

NORMAL IS 0 0 1

SURFACE AREA = 5.0000e-01 M**2

POTENTIAL = -1.9123e+01 VOLTS

UNDERLYING CONDUCTOR NUMBER IS 1

UNDERLYING CONDUCTOR POTENTIAL = -2.3562e+01 VOLTS

DELTA V = 4.4383e+00 VOLTS

INTERNAL FIELD STRESS = 3.4947e+04 VOLTS/METER

EXTERNAL ELECTRIC FIELD = -2.0598e+02 VOLTS/METER

FLUXES IN AMPS/METER**2

INCIDENT ELECTRONS	-2.0142e-05
RESULTING SECONDARIES	7.0580e-06
RESULTING BACKSCATTER	3.4652e-06
INCIDENT IONS	2.0489e-05
RESULTING SECONDARIES	0.0000e+00
BULK CONDUCTIVITY	3.4947e-12
HOPPING CURRENT	0.0000e+00
PHOTOCURRENT	0.0000e+00

TOTAL FLUX THROUGH SURFACE	1.0870e-05
----------------------------	------------

SINGLE command or MODE set >> latest

LATEST command or MODE set >> list 1 25

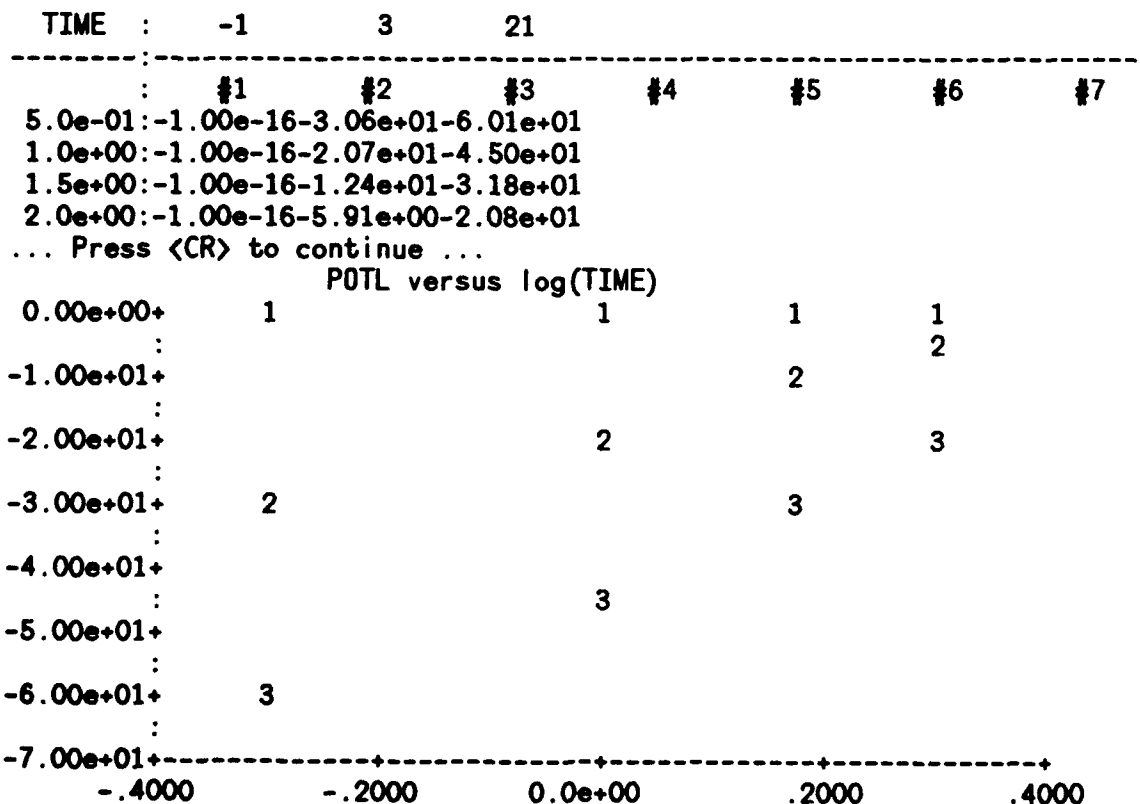
POTL IN VOLTS FOR POLAR CYCLE 4 ... TIME = 2.00e+00 SEC

1-5.91e+00	2-5.67e+00	3-5.91e+00	4-5.26e+00	5-5.36e+00
6-5.26e+00	7-4.63e+00	8-4.83e+00	9-4.63e+00	10-2.36e+01
11-2.36e+01	12-2.36e+01	13-2.36e+01	14-2.36e+01	15-2.36e+01
16-2.36e+01	17-2.36e+01	18-1.91e+01	19-1.92e+01	20-1.91e+01
21-2.08e+01	22-1.91e+01	23-2.08e+01	24-2.27e+01	25-2.28e+01

LATEST command or MODE set >> history

HISTORY command or MODE set >> -1 3 21

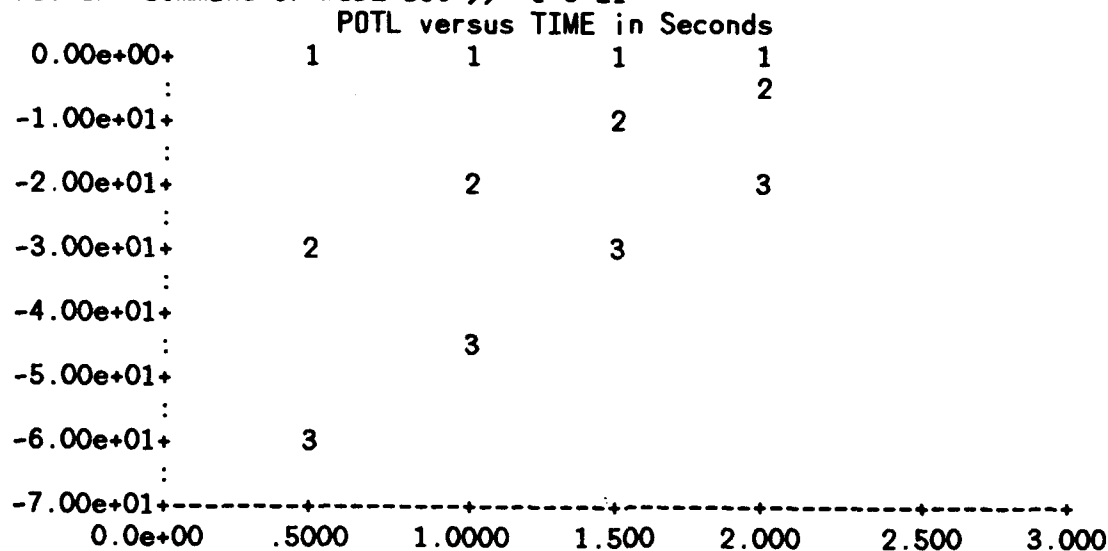
POTL in Volts



HISTORY command or MODE set >> graph
MODE RESET

HISTORY command or MODE set >> linear
MODE RESET

HISTORY command or MODE set >> -1 3 21



HISTORY command or MODE set >> subset kapt
DEFINITION OF NEW SUBSET NAMED KAPT
27 REMAINING IN GROUP

SUBSET command please >> matl kapton
18 REMAINING IN GROUP

SUBSET command please >> shape square
10 REMAINING IN GROUP

SUBSET command please >> which
MEMBERS OF GROUP KAPT
2 4 5 6 8 19 21 22 23 25
10 REMAINING IN GROUP

SUBSET command please >> done
GROUP KAPT WITH 10 MEMBERS IS NOW DEFINED
RETURNING TO MODULE 'HIST'

HISTORY command or MODE set >> special

SPECIAL command or MODE set >> noterm
MODE RESET

SPECIAL command or MODE set >> history

HISTORY command or MODE set >> group kapton
... Press <CR> to continue ...

HISTORY command or MODE set >> exit
Would you like a Laser Printer Copy? >> yes
-EXIT TRMTLK-

6.86 INTERNAL DOCUMENTATION

This is a computer printout of the actual on-line documentation available in TRMTLK. First is the complete output of the OUTLINE aid. Second is a listing of all available help sequences.

Welcome to POLAR TRMTLK ...

Any AID may be called from any MODULE

MODULES	AIDS
*****	*****
HISTORY	AGAIN
LATEST	HELP
SINGLE	LOCATION #
SPECIAL	OUTLINE
	SUBSET
	EXIT
	SUBSET [GROUP NAME]

Enter any MODULE/AID name or 'HELP' for help >> outline

Available OUTLINES:

- 1) MODULE AND AID NAMES
- 2) HISTORY
- 3) LATEST
- 4) SINGLE
- 5) SPECIAL
- 6) SUBSET
- 7) ALL OF THE ABOVE
- 8) CURRENT MODULE ONLY

Pick a number or <CR> to return to MAIN >> 7

Any AID may be called from any MODULE

MODULES	AIDS
*****	*****
HISTORY	AGAIN
LATEST	HELP
SINGLE	LOCATION #
SPECIAL	OUTLINE
	SUBSET
	EXIT
	SUBSET [GROUP NAME]

... Press <CR> to continue ...

HISTORY MODULE

---- MODES ----	---- COMMANDS ----
FLUX - FIELD - POTL	#
- DELTA - STRESS	#, #, #, ...
CYCLE # TO #	GROUP [GROUP NAME]
TIME # TO #	
TABLE - GRAPH - BOTH	
LINEAR - LOG - NUMCYCLE	

LATEST MODULE

---- MODES ----	---- COMMANDS ----
FLUX - FIELD - POTL	GROUP [GROUP NAME]
- DELTA - STRESS	LIST # TO #
SEQUENTIAL - MAGNITUDE	ALL
- ABSMAG	

... Press <CR> to continue ...

SINGLE MODULE

---- MODES ----	---- COMMANDS ----
EVERYTHING - NOTHING	#
(ALSO or NOT) followed by:	(Only command for
NUMBER STRESS	SINGLE is to enter a
CENTER NORMAL	single CELL NUMBER)
MATL SHAPE	
POTL CODE	
FLUX CNUMB	
FIELD CPOTL	
DELTA SUMMAR	

SPECIAL MODULE

---- MODES ----	---- COMMANDS ----
TERMPT - NOTERM	CYCSET #

... Press <CR> to continue ...

SUBSET SPECIFICATIONS

HELP	XLIM # TO #
INSULATOR	YLIM # TO #
BARE	ZLIM # TO #
DIRECTORY	NORMAL # # #
WHICH	NUMBERS # TO #
DONE	INCLUDE #, #, ...
OMIT	CNUMB #
	EXCLUDE #, #, ...

MATL [Material name]
 SHAPE [SQUARE, RECTAN, RIGHT, EQUIL]
 OR [Group name]
 AND [Group name]
 COMPL [Group name]
 NAME [any word]

Enter any MODULE/AID name or 'HELP' for help >> help

HELP is at hand :

- | | |
|--------------------------------------|-----------------------|
| (1) BASIC USE | (7) ERROR MESSAGES |
| (2) CURRENT MODULE | (8) COORDINATE SYSTEM |
| (3) NUMBERING CONVENTIONS | (9) LINE PRINTER FILE |
| (4) FLUX, FIELD, POTL, DELTA, STRESS | (10) DEFAULT MODES |
| (5) SUBSET AND GROUPS | (11) COMPLAINTS |
| (6) AIDS | |

Pick a number or <CR> to return to MAIN >> 1

-- BASIC USE --

TRMTLK consists of a set of MODULES, each giving the user access to a certain type of POLAR information.

'HISTORY' gives the CHARGING HISTORY of surface cells

'LATEST' gives information only from the LATEST charging cycle

'SINGLE' gives geometrical & other information about an INDIVIDUAL CELL.

'SPECIAL' performs printout control and cycle resetting.

Whichever MODULE you are in, you have available a set of COMMANDS and MODES. COMMANDS initiate output. MODES alter the form of succeeding output.

At any time instead of typing a COMMAND or MODE, you may change MODULE or call for AID. All together you have 4 choices: type a COMMAND, a MODE, a MODULE name, or an AID name.

As an example, you are in MODULE SINGLE and you type the number 317. This is a COMMAND. The terminal prints out some information (eg. potential) about the cell #317. Now you want more. You type 'EVERYTHING' (a MODE) and on the next line '317' again. More information comes out. To CHANGE to HISTORY module, you type 'HISTORY'. Now you don't know what to do, so you type 'HELP' and you get the help menu again.

For any COMMAND, MODE, MODULE, or AID, TRMTLK recognizes an entire word based on the first FOUR letters only.

... Pick another #, type 'MENU', or <CR> to return to MAIN >> 2

-- MAIN --

If you are in module MAIN, you have just entered TRMTLK I am waiting for you to choose a MODULE. All of the AIDS are available as well. Here is a list:

Any AID may be called from any MODULE

MODULES	AIDS
*****	*****
HISTORY	AGAIN
LATEST	HELP
SINGLE	LOCATION #
SPECIAL	OUTLINE
	SUBSET
	EXIT
	SUBSET [GROUP NAME]

... Pick another #, type 'MENU', or <CR> to return to MAIN >> 3

-- NUMBERING CONVENTIONS --

TRMTLK has particular NUMBERING CONVENTIONS for SURFACE CELLS and CONDUCTORS. A POLAR OBJECT is composed of N surface CELLS and L CONDUCTORS. NEGATIVE numbers (-L through -1) refer to CONDUCTORS. The POSITIVE numbers (1 through N) refer to surface CELLS. These are the small RECTANGLES and triangles covering the spacecraft.

... Pick another #, type 'MENU', or <CR> to return to MAIN >> 4

-- FLUX FIELD POTL DELTA STRESS --

These are the 5 DYNAMIC pieces of information pertaining to any surface CELL. They CHANGE each TIME CYCLE. Only FLUX and POTL pertain to CONDUCTORS.

- FLUX - TOTAL ELECTRIC FLUX to a CELL or CONDUCTOR due to INCIDENT PARTICLES, ELECTRON BACKSCATTER, HOPPING CURRENT and and PHOTOEMISSION. UNITS are AMPS/METER**2.
- FIELD - the EXTERNAL ELECTRIC FIELD in the volume immediately above a surface CELL. UNITS of VOLTS/METER.
- POTL - ELECTRIC POTENTIAL on the EXTERNAL surface of a CELL or a CONDUCTOR. UNITS are VOLTS.
- DELTA - DIFFERENCE between surface POTENTIAL of a DIELECTRIC CELL and POTENTIAL of the UNDERLYING CONDUCTOR. UNITS are VOLTS.
- STRESS - INTERNAL ELECTRIC FIELD STRESS for DIELECTRIC CELLS. Proportional to DELTA and inversely proportional to MATERIAL THICKNESS. UNITS are VOLTS/METER.

... Pick another #, type 'MENU', or <CR> to return to MAIN >> 5

-- SUBSET and GROUPS --

You can use SUBSET to DEFINE a GROUP of CELLS that are of special interest to you. For instance, you might want all KAPTON coated surfaces, or all the cells over conductor #2, or the intersection of those two sets.

You enter subset by typing 'SUBSET', or 'SUBSET' followed by a group name. You may define up to 34 different groups, in addition to the 2 default groups ALL and NULL (complete set and empty set).

The SUBSET specs can be classified into NARROW DOWN TYPES, SET OPERATIONS, INFORMATION REQUESTS, and OTHERS.

... Press <CR> to continue ...

***** NARROW DOWN *****

A new SUBSET consists of all CELLS and CONDUCTORS. The NARROW DOWN specifications eliminate any cells that don't fit. All of the NARROW DOWN specifications, except 'NUMBERS', eliminate all CONDUCTORS, -1 through -7.

- XLIM # TO # - EXCLUDE cells with CENTERS outside of given X-axis limits.
- YLIM # TO # - similar to XLIM
- ZLIM # TO # - similar to XLIM
- INSULATOR - EXCLUDE CONDUCTING cells
- BARE - EXCLUDE INSULATING cells

NORMAL # # # - EXCLUDE any cell with a surface NORMAL other than this.
 CNUMB # - EXCLUDE cells with UNDERLYING CONDUCTORS other than this.
 MATL [material] - EXCLUDE cells NOT of this MATERIAL
 SHAPE [shape] - EXCLUDE any cell NOT in this SHAPE
 NUMBERS # TO # - EXCLUDE if CELL NUMBERS are NOT within these LIMITS.
 if 1st number < -7, NO CONDUCTORS will be eliminated.

... Press <CR> to continue ...

***** SET OPERATIONS *****

These allow you to COMBINE the CURRENT GROUP with any PREVIOUSLY DEFINED GROUP. To start over on CURRENT GROUP, 'OR ALL' or 'COMPL NULL' brings back all members, while 'AND NULL' or 'COMPL ALL' eliminates all. The GROUP named (except ALL and NULL) must have been defined by the user.
 OR [name] - present GROUP is UNION of PRESENT GROUP and NAMED GROUP.
 AND [name] - present GROUP is INTERSECTION of PRESENT GROUP and NAMED GROUP.
 GROUP. COMPL [name] - present GROUP is COMPLEMENT of NAMED GROUP.

***** INFORMATION *****

WHICH - prints out CURRENT GROUP MEMBERS
 DIRECTORY - gives LIST of PREVIOUSLY DEFINED GROUPS
 HELP - this message

***** OTHER SPECS *****

NAME [name] - CHANGE the NAME of CURRENT GROUP to this
 INCLUDE [#'s] - up to 14 specified CELLS and CONDUCTORS become GROUP MEMBERS.
 EXCLUDE [#'s] - up to 14 MEMBERS REMOVED from GROUP
 DONE - CATALOGUE this GROUP for later use. Return to MODULE.
 OMIT - DON'T CATALOGUE this GROUP. Return to MODULE.
 ... Press <CR> to continue ...

SUBSET SPECIFICATIONS

 HELP XLIM # TO #
 INSULATOR YLIM # TO #
 BARE ZLIM # TO #
 DIRECTORY NORMAL # # #
 WHICH NUMBERS # TO #
 DONE INCLUDE #, #, ...
 OMIT CNUMB #
 EXCLUDE #, #, ...
 MATL [Material name]
 SHAPE [SQUARE, RECTAN, RIGHT, EQUIL]
 OR [Group name]
 AND [Group name]
 COMPL [Group name]
 NAME [any word]

... Pick another #, type 'MENU', or <CR> to return to MAIN >> 6

-- AIDS --

AIDS are available from any MODULE. You CANNOT generally call one AID while using another. You CANNOT call 'EXIT' while using HELP for example.

AGAIN - REPEAT the most recent PROMPT
 HELP - prints various informative MESSAGES
 LOCAT # - gives the coordinates of the CENTER of the indicated CELL.
 OUTLINE - prints a MENU, a LIST of available MODULES, COMMANDS, MODES, AIDS.
 SUBSET - allows user to DEFINE a GROUP of CELLS and CONDUCTORS that are of particular interest. The entire set of CELLS is NARROWED DOWN to a desired SUBSET.

SUBSET [group name] - if the GROUP NAME has been previously DEFINED, permits ALTERATION of the members. If not, a NEW GROUP with that NAME is then defined.

EXIT - ends TRMTLK use.

... Pick another #, type 'MENU', or <CR> to return to MAIN >> 7

-- ERROR MESSAGES --

Let's say you did something I didn't expect, or I just got confused. I will print out an ERROR MESSAGE. The 1st word of an error MESSAGE has ASTERISKS around it, and identifies the SUBROUTINE where the problem occurred, for example '*** SINGLE ***'. What do you do then? Try it again, or try something else. You can't damage the program. Or call for 'OUTLIN' or 'HELP'.

... Pick another #, type 'MENU', or <CR> to return to MAIN >> 8

-- COORDINATE SYSTEM --

The COORDINATE SYSTEM used by TRMTLK is in GRID UNITS

... Pick another #, type 'MENU', or <CR> to return to MAIN >> 9

-- LINE PRINTER FILE --

TRMTLK creates a LINE PRINTER FILE. At the end of a run, you can send this FILE to the PRINTER to make a hard copy. If you are using a slow terminal, you can ask for MODE 'NOTERM' in MODULE 'SPECIAL'. In this MODE, all output goes to the line printer file only. You can still get a complete hard copy at the end of the run.

... Pick another #, type 'MENU', or <CR> to return to MAIN >> 10

-- DEFAULTS --

At the beginning of TRMTLK, all MODES are set to a DEFAULT value. Default MODES for each MODULE are:

HISTORY: POTL, ALL CYCLES, BOTH, LOG
 LATEST : POTL, SEQUENTIAL
 SINGLE : ALSO NUMBER, CENTER, MATL, POTL
 NOT (everything else)
 SPECIAL: TERMPRT

... Pick another #, type 'MENU', or <CR> to return to MAIN >> 11

-- COMPLAINTS --

TRMTLK was written to be a convenient, user-oriented DATA REDUCTION tool. If you don't like it, or would like to see some changes, please write to:

SYSTEMS, SCIENCE, and SOFTWARE

P.O. Box 1620

La Jolla, California 92038.

Mark it 'Attention: John Lilley'.

... Pick another #, type 'MENU', or <CR> to return to MAIN >>

Returning to MODULE MAIN

Enter any MODULE/AID name or 'HELP' for help >> history

HISTORY command or MODE set >> help

HELP is at hand :

- | | |
|--------------------------------------|-----------------------|
| (1) BASIC USE | (7) ERROR MESSAGES |
| (2) CURRENT MODULE | (8) COORDINATE SYSTEM |
| (3) NUMBERING CONVENTIONS | (9) LINE PRINTER FILE |
| (4) FLUX, FIELD, POTL, DELTA, STRESS | (10) DEFAULT MODES |
| (5) SUBSET AND GROUPS | (11) COMPLAINTS |
| (6) AIDS | |

Pick a number or <CR> to return to HIST >> 2

The COMMANDS for module HISTORY are CELL NUMBERS. You can type ONE cell number, or a string of up to 15 numbers on the same line. Conductor numbers (negative) are also valid. If you have a set of numbers you plan to use more than once, you can define them as a group using SUBSET.

Your output will be a TABLE and/or a GRAPH of whichever one of the 5 quantities of the current mode. Notice that the HISTORY modes are distinct from the LATEST modes. If you specify a range of cycles or a range of times, only those boundaries will be included in the TABLE/GRAPH. The X-axis of the GRAPH will be the TIME, log base 10 of TIME, or the CYCLE numbers -- depending on whether the MODE is LINEAR, LOG, or NUMCYC.

HISTORY MODULE

---- MODES ----

FLUX - FIELD - POTL
- DELTA - STRESS

CYCLE # TO #

TIME # TO #

TABLE - GRAPH - BOTH

LINEAR - LOG - NUMCYCLE

---- COMMANDS ----

#

#, #, #, ...

GROUP [GROUP NAME]

... Pick another #, type 'MENU', or <CR> to return to HIST >>

Returning to MODULE HIST

HISTORY command or MODE set >> latest

LATEST command or MODE set >> help

HELP is at hand :

- | | |
|--------------------------------------|-----------------------|
| (1) BASIC USE | (7) ERROR MESSAGES |
| (2) CURRENT MODULE | (8) COORDINATE SYSTEM |
| (3) NUMBERING CONVENTIONS | (9) LINE PRINTER FILE |
| (4) FLUX, FIELD, POTL, DELTA, STRESS | (10) DEFAULT MODES |
| (5) SUBSET AND GROUPS | (11) COMPLAINTS |
| (6) AIDS | |

Pick a number or <CR> to return to LATE >> 2

LATEST prints information about a GROUP of CELLS and CONDUCTORS.

The 3 COMMANDS are 3 ways to specify a GROUP:

'ALL' will include the full set of CELLS and CONDUCTORS

'GROUP' followed by a GROUP name prints only those in the defined group

'LIST' followed by 2 numbers will list the intervening cells. The information printed can be FLUX, FIELD, POTL, DELTA, OR STRESS.

In SEQUENTIAL MODE, the information will be ordered by CELL number.

In MAGNITUDE MODE, it will be ordered by the VALUES being printed.

In ABSMAG MODE, the ordering will be by ABSOLUTE VALUES

LATEST MODULE

---- MODES ----

FLUX - FIELD - POTL

- DELTA - STRESS

SEQUENTIAL - MAGNITUDE

- ABSMAG

---- COMMANDS ----

GROUP [GROUP NAME]

LIST # TO #

ALL

... Pick another #, type 'MENU', or <CR> to return to LATE >>

Returning to MODULE LATE

LATEST command or MODE set >> single

SINGLE command or MODE set >> help

HELP is at hand :

- | | |
|--------------------------------------|-----------------------|
| (1) BASIC USE | (7) ERROR MESSAGES |
| (2) CURRENT MODULE | (8) COORDINATE SYSTEM |
| (3) NUMBERING CONVENTIONS | (9) LINE PRINTER FILE |
| (4) FLUX, FIELD, POTL, DELTA, STRESS | (10) DEFAULT MODES |
| (5) SUBSET AND GROUPS | (11) COMPLAINTS |
| (6) AIDS | |

Pick a number or <CR> to return to SING >> 2

-- SINGLE --

This MODULE gives various information about a SINGLE SURFACE. The only COMMAND is the CELL NUMBER.

MODES specify which cell PROPERTIES are to be printed. If you want all of them, set 'EVERYTHING'. To wipe the slate clean, type 'NOTHING'. If you want to INCLUDE a cell property, type 'ALSO' and the property name, eg. 'ALSO POTL'. To EXCLUDE a property, use 'NOT' in place of 'ALSO'.

The DYNAMIC PROPERTIES, which change from one cycle to the next, are 'FLUX', 'FIELD', 'POTL', 'DELTA', 'SUMMAR' and 'STRESS'.

The STATIC PROPERTIES are:

NUMBER - the cell number
 CENTER - coordinates of the cell center.
 MATL - material name of this surface.
 NORMAL - normal vector pointing out of this surface cell.
 SHAPE - surface shape. (SQUAre, RIGHT triangle, RECTangle, or EQUilateral triangle).
 CNUMB - conductor number of the underlying conductor
 CPOTL - potential of the underlying conductor
 ... Press <CR> to continue ...

SINGLE MODULE

----	MODES	----	COMMANDS	----
EVERYTHING	- NOTHING		#	
(ALSO or NOT)	followed by:		(Only command for	
NUMBER	STRESS		SINGLE is to enter a	
CENTER	NORMAL		single CELL NUMBER)	
MATL	SHAPE			
POTL	CODE			
FLUX	CNUMB			
FIELD	CPOTL			
DELTA	SUMMAR			

... Pick another #, type 'MENU', or <CR> to return to SING >>
 Returning to MODULE SING

SINGLE command or MODE set >> special

SPECIAL command or MODE set >> help

HELP is at hand :

- | | |
|--------------------------------------|-----------------------|
| (1) BASIC USE | (7) ERROR MESSAGES |
| (2) CURRENT MODULE | (8) COORDINATE SYSTEM |
| (3) NUMBERING CONVENTIONS | (9) LINE PRINTER FILE |
| (4) FLUX, FIELD, POTL, DELTA, STRESS | (10) DEFAULT MODES |
| (5) SUBSET AND GROUPS | (11) COMPLAINTS |
| (6) AIDS | |

Pick a number or <CR> to return to SPEC >> 2

-- SPECIAL --

This could be called MODULE miscellaneous. Presently, it has only 2 functions: PRINT CONTROL and CYCLE RESETING.

By default, all printout goes BOTH to the terminal and to file 3. If you are generating a lot of printout and don't want to wait for it at the the terminal, use mode 'NOTERM'. Then the output will go only to file 3, which can be sent to the line printer upon exiting TRMTLK. You use CYCLE RESET if you are going to make further POLAR RESTART runs, but have no further use for the data already generated. In SPECIAL, if you type 'CYCSET 0', all old data will be discarded from the HISTORY files. LATEST and SINGLE will remain unchanged.

SPECIAL MODULE

---- MODES ----

TERMPT - NOTERM

---- COMMANDS ----

CYCSET #

... Pick another #, type 'MENU', or <CR> to return to SPEC >>
Returning to MODULE SPEC

SPECIAL command or MODE set >> exit
Would you like a Laser Printer Copy? >> yes
-EXIT TRMTLK-

6.90 Source Code Maintenance

The source code for POLAR is normally divided up into separate directories. Each directory contains the source code needed to create the library or executable it is named after. Currently, the main development effort for the POLAR package takes place in an UNIX environment. A specialized shell script (a file containing operating system commands) is used to set up, then execute the utility program, make. The instructions used by make to put together libraries and executable programs reside in each directory and are called "Makefile".

The sections which follow contain information on various aspects of the compilation process. These files can also be found in the key locations in the POLAR source code directory structure, as described in the text below. Additional, more specific information can be found in the Makepl and all of the Makefile files.

6.91 Installing the Source Code

This file is called "Install.notes" and can be found in the top POLAR source code directory (if it has been installed already).

---Install.notes

Here are some useful commands for installing Polar 1.3.
(UNIX commands are indicated by a leading ">")

Moving polar 1.3 to a new machine.

From a tar format tape.

- a) Create a directory to be used as the top of the polar tree, a good name might be "pol1.3". There should be enough disk space for about 25 to 30 Megabytes of data. This can be reduced after installation.
- b) Go to the top of the polar tree.
>cd pol1.3

- c) Read the tape, replace TAPE_DEV with the name of the tape device containing the tape.
`>tar xvfo - . < TAPE_DEV`
- d) Edit the Makep1 file and change CPU_TYPE to the new cpu type and P1_NODE the full path name to the directory name used in step (a).
- e) Now make a version of Polar 1.3 for the new machine.
 To make a new version of Polar 1.3, type:
`>Makep1`

Between networked UNIX machines, but no NFS mounts.

- a) Log into the machine which presently has Polar 1.3.
- b) Go to the top of the polar tree.
`>cd pol1.3`
- c) Use tar to send the old copy to the new location. In the command below, replace TARGET_HOST with the name of the machine receiving Polar 1.3 and WHERE with the name of the directory on TARGET_HOST which will contain the new Polar 1.3.

on Ridge 3200's

```
>tar cf - ./pol1.3 | rshell TARGET_HOST "cd WHERE; tar xfo -"
```

on other UNIX systems

```
>tar cf - ./pol1.3 | rsh TARGET_HOST "cd WHERE; tar xfo -"
```

- d) Now log into the receiving machine and go to the new pol1.3 directory. Edit the Makep1 file and change CPU_TYPE to the new cpu type and P1_NODE the full path name to the new pol1.3 directory.

- e) Now make a version of Polar 1.3 for the new machine. To make a new version of Polar 1.3, type:

>Makep1

Cleaning up intermediate files after installation.

Go the top of the polar tree and enter

> Makep1 clean

To get rid of everything created by the make'ing process (but no source code), enter

> Makep1 clobber

6.92 General Information for the Main Source Code Version

This file is called "README" and can be found in the top POLAR source code directory.

---README

README file for Polar 1.3 Main node make structure 4/21/89

Overview

The goal of the current compilation structure is to create a system independent method of keeping the POLAR executables up to date. The UNIX utility, make, is used to check the various subdirectories to ensure the target library or executable is current with respect to any modifications in the parts which are used to build it. There are two separate structures in POLAR, one is called the Main node and the second is called the User node. The Main node is the structure which is distributed to new sites. It is the version which is maintained off site and is considered to be constant. If bugs are found or when improvements are made, the changes are made assuming everyone has the same Main version of Polar 1.3. The User node is designed to be used as the area where changes can be made by the local users. This file discusses the use of the Main node make structure. For a discussion of the User node make

structure, please look in the User subdirectory in the README file found there.

Getting Started

The first step is to install POLAR. Instructions for installing Polar are found in the file, Install.notes. The contents of this file can also be found in the Polar User's Manual.

Make Structure

The Polar package is organized as a self-contained directory structure. At the top directory or main node, are several documentation files (README, Install.notes, and Make.info), the source code directories (genlib, pollib, nterak, orient, vehicl, vellib, shontl and utils), the include file directory (inc), the directory containing the object code libraries and executable files (archives and bin, respectively), a directory structure designed for development work (User), a file whose name represents the current version of Polar (presently, P_VER1.3.0), and finally the files and scripts used to check and maintain the source code (Makefile and Makep1).

The Make structure is controlled by the Makep1 script. This script defines shell environment variables and machine dependency flags which are sent to the make utility. Each of the source code directories contain a Makefile which instruct make which and how items need to made in that directory. There is also a Makefile in the top directory which knows the order in which to make each directory. The Makefile files contain instructions that behave like command keywords.

These commands are defined in the top node Makefile and in each source code node Makefile. For more information on the main node make structure, please see the comments in the main node "Makep1" file.

The User subtree should be used to create modified versions of Polar. In addition to the instructions contained in the main node make structure, the user make structure knows how to get versions of source code from the appropriate location in the main version and how to use as much compiled code from the main version. For more information on the user node make structure, see the "README.User" file in the top User node directory (named "User" in the main node directory).

Making Changes

Any supported changes to the main version of Polar will be accompanied with instructions. Of course, no one is watching so you can still change the Polar source code in the main version. It is not a good idea. If changes are desired, a better place to do it is in a User node.

The exception to this would be if an include file needs to be changed. For example, if the NTERAK module which calculates sheath wakes runs out of space because its data arrays are too small, the Shado.h file needs to be modified. To do this, go to the "inc" directory. Edit the file, Shado.h and increase the parameter, msurf, to the desired number. When the file is saved, the date of last modification is changed. Now return to the main directory and type "Makep1 check". This command directs make to traverse through the entire Polar source code structure and update any files, libraries, or executables affected by the change. Unfortunately, anyone using the User node structure must also update their versions of Polar.

The necessary instructions are discussed in the README in the User directory.

In any case, if the main version of Polar is modified, the version variable in Makepl should be changed to something appropriate.

Porting to Different Machines and Other Ugly Tasks

Currently, Polar 1.3 is able to run on a Sun 3 and on a Ridge 3200. The previous versions of Polar has been ported to VAX/VMS, CRAY/CDS, and other even more obscure operating environments (UNIVAC/EXEC 8, CYBER). The coding standards used developing the Polar package prevent porting troubles, i.e. built in system dependencies.

Moving Polar to other UNIX should be fairly straightforward. If the new machine is not a Ridge 3200 (running RX/V) or a Sun 3, try the Sun 3 flags first. The Sun 3 flags favor more standardish Fortran and UNIX usages. The file "Install.notes" contains some useful hints on moving the source code. The usual routines which may require modification are found in genlib. The different, system dependent routines are identified by flags added to the source code name. For example, uflset.c_SUN3 is the Sun 3 version of uflset.c and uflset.c_R3200 is the Ridge 3200 version.

The routines which may require attention when moving to a UNIX environment are segkil.c (floating point error handling), uflset.c (floating point underflow error handling), iclock.c (time and date information), and mclock.c (run timing information).

When porting to operating systems other than UNIX, setchr.f (word size, byte size, and character size information) may need modification and all of the c routines may need to be replaced with Fortran equivalents (see the _VAX routines).

The Makepl structure is designed to be used in an NFS environment with different types of CPUs using the same disk storage. It is possible to use the same set of source to maintain current versions of the executables in order to reduce disk usage and time spent maintaining different versions of the source.

Tidbits about Makefiles

See the file, Make.info. There should be online information available on all UNIX systems via the "man make" command. Additional information can be found in the system manual titled Programming Tools or Programming Utilities.

Useful References

Makepl (the file with the UNIX commands used to run make) Polar 1.3 User's Manual

Helpful People

David Cooke AFGL, (617) 377-2931
John Lilley S-CUBED, (619) 453-0060
email =>lilley@scubed.scubed.com

6.93 Information for Making Local or User Modified Versions of POLAR

This file is called "README" and is found in the subdirectory, "User" beneath the top POLAR source code directory. The directory, User, contains the Makep1 and Makefile files necessary to create versions of POLAR which are independent of the Main version.

---README.User

README file for Polar 1.3 User node make structure 4/21/89

Overview

The User make structure should be used to test modifications to the Polar source code. By keeping the distributed source code and the local modifications separated, it will be easier to update the Polar package while retaining any desired changes.

Getting Started

(UNIX commands are indicated by a leading ">")

The first step is to copy the prototypical User directory found in the main node to a working directory location. After creating and changing directories to the new user node directory (called USER_NODE from now on), enter the following command to copy the User structure. Replace WHERE with the name of the top of the main Polar source directory.

```
>(cd WHERE; tar cf - . ) | tar xfo -
```

Cool, now the whole Polar directory structure complete with customized Makefile files and Makep1 script has been copied to your working area. To complete the installation, two variables need to be modified in the Makep1 file. This is a shell script which sets machine dependent flags and defines directory locations.

Find the line which begins with "P1_NODE=". Place the complete path name to the main node directory (same as WHERE above) immediately after the "=" without any spaces. This is used to find the main versions of source and object code and archive libraries.

Now find the line "NODE=". Put the name of the top user directory (same as USER_NODE above) immediately after the "=". This defines the locations of your modification directories.

One more thing to check. Find the line beginning with "CPU_TYPE=". This defines a flag which is used to make machine dependent decisions. Currently supported Polar versions are defined for the CPU_LIST variable.

Some more options and flags can be set, take a look at the comments at the top of the Makep1 file find out more about them.

After making the desired changes in Makep1, it is necessary to complete the initialization of the user area. If most or all of the directories will be changed, from the USER_NODE directory enter "Makep1 install_user". If only a few directories are going to be modified, the install_user command can be issued in just those directories.

If the USER_MODE flag is changed after a user node has been installed, it will need to be reinstalled. The best way is to enter:

```
>Makep1 clobber
```

```
>Makep1 install_user
```

This will replace the old make products and flags with the new ones.

Making Changes

Here are some quick notes for doing some of the standard things to a User version.

1) Grabbing source code from main version

To grab the main version of the source, just call Makep1 with the name of the source code desired. For example, to change the optin.f routine in nterak, enter "../Makep1 optin.f" from the nterak directory.

2) Things which will may need to be modified

a) Makep1

Modify Makep1 if a new directory is added, a new USER_MODE or CPU_TYPE is added, or a different Polar library is desired.

b) main Makefile

The top Makefile should be modified if a new executable or library is added or a new make option is added to all or some of the subdirectories.

c) changing Polar libraries to be used

The Polar libraries used for linking the executables can be set in two places. If a library is to be used for all of the executables in the structure, change the Makep1. If a library is desired for just one executable, modify the Makefile in that executable's directory.

d) where do executables go?

This is determined in the Makep1 file by the BIN_DIR flag. The default is in NODE/bin. There is a similar variable for the location of the user versions of archive libraries (ARC_DIR) in Makep1. The particular Makefile for the executable can also be modified.

3) Modified routines

Start by grabbing the main version of the source. For example, to change the optin.f routine in nterak, enter `"../Makep1 optin.f"` from the nterak directory. When you're ready to compile and link the new version, modify the Makefile (note 5 below) so that it will use the local version.

4) New routines

Just add the name of the routine to the appropriate local source code list.

5) modify Makefiles in directories with changes

There are some notes in the Makefile files which explain the basic changes needed to use the modified version of a routine. Basically, all source code files in a given directory are listed by name. The _FSRC kinds of variables contain the list of Fortran routines. There are two lists for each type of source code. One is the list of routines which are taken from the main Polar node. The other is the list of local routines. A subroutine name can only appear in one list. So the standard thing to do is to move the modified routine from the main node list to the local list. If any new include file dependencies are added (or removed), the individual dependency instructions at the end of the Makefile should be updated too.

6) Moving routines from one directory to another

Take the name of the routine out of the source code list in the Makefile and put it in the equivalent list in the new directory.

If there are any applicable Makefile instructions for file dependencies or special handling instructions, move them too. Oh, and don't forget to move the routine.

7) include files

Only change include files in the inc directory, otherwise bugs may arise when object code from the different directories is linked together. When modifying include files, it is a good idea to use the OLD_WAY flag for USER_MODE in Makep1. This may require re-install_user'ing the desired directories.

Useful References

README and Make.info in Polar 1.3 Home directory

Makep1

Polar 1.3 User's Manual

Helpful People

David Cooke AFGL, (617) 377-2931

John Lilley S-CUBED, (619) 453-0060

email =>lilley@scubed.scubed.com

6.94 A Quick Summary of Makefile Syntax

The make utility utilizes its own language to define the relationships between files and the methods to create necessary files. The following file is called "Make.info" and can be found in the top POLAR directory. For further information, on line help is available via "man make" and the documentation for most UNIX systems contains a helpful document in the "Programming Utilities" reference under the topic, make.

---Make.info

Here are some helpful, hopefully, notes on Makefile syntax.

Makefile instruct make how to check the currentness of the parts needed to make the target item.

To use make with these Makefiles, first edit ../Makep1 to make sure it set up properly for this machine. Then instead of typing "make <options>", type "../Makep1 <options>". This will define the necessary variables and then automatically call "make <options>".

How to add a new routine the Makefile in the appropriate directory:

Add the name of the new routine to the _CSRC or _FSRC list. If it contains an include file, add a dependency line at the bottom of the file.

Some notes on make syntax:

Variables are defined using VARIABLE_NAME = stuff

Rules are defined using TARGET: DEPENDENCIES

The first rule whose target does not start with a "." is the default action. ("check" in this file.)

Commands which start with "@" are not printed when executed.

Errors on commands starting with "-" are ignored.

Variables are of the form \$(VARIABLE_NAME). If the variable name is a single character, such as H (for HERE), it can be written as \$H. make will figure out some useful wildcards. The ones used in these Makefiles are:

`$*` = the file part of the target with the suffix removed.
`$0` = the full file name of the target being evaluated.
`$(0F)` = the full file name with the directory portion removed.
`$(0D)` = the directory portion of the full file name.
`$(FSRC:.f=.o)` = The suffices of items listed in FSRC are changed from ".f" to ".o".

(See also the UNIX manual entry for make. Some manuals also have a helpful paper on make in the Programming Tools or Utilities manual which is not online.)

setting MAKE variable to make allows the make -n option to traverse the entire make tree. So if a rule used to create something calls make, that make is also tested. The -n option is used to print out the commands make will generate without executing them.